

AD-A041 626

ROCKWELL INTERNATIONAL CEDAR RAPIDS IOWA COLLINS RA--ETC F/G 17/7
DIGITAL FLIGHT DIRECTOR COMPUTER.(U)
APR 75 J P DESMOND, G E FORQUER

UNCLASSIFIED

ASD-TR-76-3

F33657-73-C-0120
NL

1 of 4
ADA041626



AD A041626

ASD-TR-76-3

12
B.G.

DIGITAL FLIGHT DIRECTOR COMPUTER

Collins Radio Group
Rockwell International
Cedar Rapids, Iowa 52402

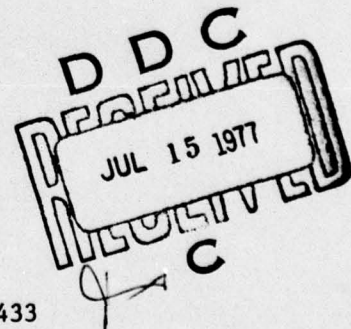
TECHNICAL REPORT ASD-TR-76-3

Final Report for Period February 1973-April 1975

Approved for public release; distribution unlimited.

AD No. _____
DDC FILE COPY

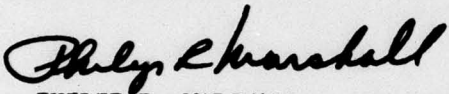
AERONAUTICAL SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433



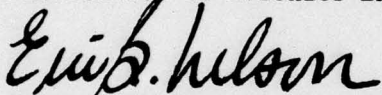
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

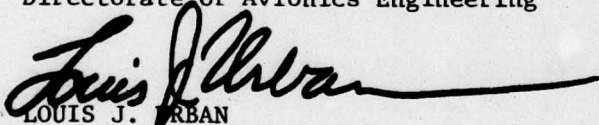
This technical report has been reviewed and is approved for publication.



PHILIP R. MARSHALL
Chief, Instruments Branch
Information Engineering Division
Directorate of Avionics Engineering



ERIC B. NELSON, Lt Col, USAF
Chief, Information Engineering Division
Directorate of Avionics Engineering



LOUIS J. URBAN
Technical Director
Directorate of Avionics Engineering

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) ASD-TR-76-3 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) DIGITAL FLIGHT DIRECTOR COMPUTER. FINAL REPORT		5. TYPE OF REPORT & PERIOD COVERED Final 2-73, 4-75
7. AUTHOR(s) (10) John P./Desmond Gary E./Forquer		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Collins Radio Group ✓ Rockwell International Cedar Rapids, Iowa 52402		8. CONTRACT OR GRANT NUMBER(s) (15) F33657-73-C-0120 New
11. CONTROLLING OFFICE NAME AND ADDRESS ASD/AEAI WPAFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 27131500 64212F
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (9) Final rept. Feb 73 - Apr 75		12. REPORT DATE (12) 45
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release, distribution unlimited. (11) Apr 75 (12) 337p.		13. NUMBER OF PAGES
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
18. SUPPLEMENTARY NOTES		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Flight Director Computer Range Estimator Digital Error Correcting Kalman Filtering Circular Capture		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This document is a comprehensive report on the Digital Flight Director program which included formulation of a universal flight director control law, software and hardware prototype development, and simulation and flight tests performed to verify the system. A		

DDC
APPROVED
JUL 15 1977
C

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 904

ASD-TR-76-3

PREFACE

This work was accomplished by the authors as employees of Collins Radio Group, Rockwell International, Cedar Rapids, Iowa 52402. The program was controlled by Aeronautical Systems Division. The project and task number is 27131500.

This report covers research done over the period of February 1973 - April 1975. It was submitted by the authors in September 1975.

The original study was done by Gerald W. Francis of the Aeronautical Systems Division. A technical report was written by Gerald W. Francis and published in August 1974 as ASD-TR-74-20.

ADDITIONAL	
RTS	Write Section <input checked="" type="checkbox"/>
ORC	Dist Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
I. Introduction	1
II. Hardware Implementation.	2
General Characteristics.	2
Interfacing Systems	2
Flight Director Inputs	2
Flight Director Outputs.	5
Block Diagram	5
Aircraft Systems Interface	8
Input Parameters	8
Input Circuitry	8
Synchro Demodulators	8
Radio Receivers.	8
Heading and Course Error Demodulators	10
Flag Receivers	10
Analog Multiplexer	10
A to D Conversion	10
Digitizing Accuracy.	10
Output Parameters	13
Output Circuitry	13
Output Control Logic	13
Output Accuracies	13
I/O Control Logic	13
Data Input	18
Data Output	18
Oscillator and Master Timer.	18
Power On Clear	22
Digital Computer	22
General Characteristics	22
Stack Operation	25
Processor Organization	30
Data Path Logic	32
Microcontroller	37
Transfer Bus	49
Memory	57
Interrupts	62
Instruction Set and Interrupt Handler	64
Self Test	76
Power Requirements	77
Power Converter	78
Configuration	79
Airspeed Controller	81
Aircraft Interconnect	84
Applicability to Mass Production	93

	<u>Page</u>
III. Control Law Design and Analysis	94
Circular Capture Algorithm	94
Kalman Estimation Process	107
TACAN Kalman Estimation Process	113
ILS Kalman Estimation Process	113
Localizer Kalman Estimation Process	115
Glideslope Kalman Estimation Process	115
Manual Heading Control Laws	118
TACAN Control Laws	118
Heading Hold Mode	119
Capture Mode	120
Track Mode	120
Overstation Mode	121
Range Estimation	121
Wind Estimation	121
ILS Localizer Control Laws	122
Heading Hold Mode	122
Capture Mode	123
Track Mode	124
Range Estimation	124
Wind Estimation	125
ILS Glideslope Control Laws	125
Precapture and Idle Modes	126
Capture Mode	126
Track Mode	126
Range Estimation	126
Wind Estimation	127
IV. Software Design	127
Self Test Mode	129
Off Mode	129
Manual Heading Mode	130
TACAN Mode	130
ILS Mode	130
Mode Control	130
Kalman Filter	131
Math Package	131
V. Simulation Studies	134
Kalman Estimator	139
Manual Heading	140
TACAN	140
ILS	141

	<u>Page</u>
ILS Localizer	141
Jet Transport - 124.5 Knots	141
Propeller Powered Aircraft - 120 Knots	142
Jet Aircraft - 150 Knots	142
ILS Glideslope	143
VI. Flight Test Results	190
Deviations from the Ideal	209
Airspeed Inputs	209
Radar Altitude	209
Roll Rate Limit	210
Appendix A System Flow Charts	211
Appendix B System Definitions.	232
Appendix C Fortran Program Listing	240
Appendix D Fortran Program Listing (Scaled)	261
Appendix E AED Program Listing (Scaled)	281
Appendix F Test Equipment	311

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1	562A-14 Digital Flight Director Computer	3
2	377B-1 Airspeed Controller	4
3	DFDC Block Diagram	6
4	Flag Receiver Action	11
5	I/O Control Signals	16
6	I/O Control Logic	17
7	Timer Interrupts	21
8	Stack Evaluation Example Operation	26
9	Stack Organization	28
10	Processor Organization	31
11	Data Path Logic	33
12	Microcontroller	38
13	Basic Microcontroller Timing	40
14	Control Store Mapping	42
15	Microprogram Field Definitions	43
16	Branch Control Logic	47
17	Transfer Bus Organization	50
18	Transfer Bus Line Definitions	51
19	Transfer Bus Timing	53
20	Transfer Bus Priority Timing	55
21	Composite Bus Request and Transfer Activity	56
22	CAPS Memory Module	58
23	Storage Module Map	59
24	PROM Power Switching	61
25	Memory Control Timing	63
26	Format of Local Environment Syllable	65
27	Program Configuration	70
28	Interrupt Handling Sequence	71
29	DFDC Installation Control Drawing	80
30	ASC Installation Control Drawing	82
31	ASC Electrical Diagram	83
32	J1 - Signal Connector	85
33	J2 - Test Connector	86
34	DFDC Input Signal Characteristics	87
35	DFDC Output Signal Characteristics	89
36	DFDC External Wiring Diagram	90
37	Aircraft Distances and Angles	95
38	ILS Localizer Geometry	96
39	ILS Glideslope Geometry	97
40	Lateral Axis Control	99
41	Basic Geometry for ILS Localizer Computations	101
42	Overview of Multistage Decision Process	103
43	Vector Triangle Computations	104
44	Originally Proposed Localizer Circular Capture Algorithm	105

LIST OF ILLUSTRATIONS (Concluded)

<u>Figure</u>		<u>Page</u>
45	Current Localizer Circular Capture Algorithm	106
46	Originally Proposed Glideslope Circular Capture Algorithm	108
47	Current Glideslope Circular Capture Algorithm	109
48	Discrete Time Kalman Estimator	110
49	Position, Position Rate (y, \dot{y}) Kalman Estimator	111
50	ILS Lateral Axis Control - Estimation Process	114
51	Beam Angle, Beam Rate ($\theta, \dot{\theta}$) Kalman Estimator	116
52	EAI Pacer 100 Digital Computer	135
53	EAI Pacer, 680 Computers, and Interface Unit	136
54	Cockpit Facility	137
55	Digital/Analog Interface for DFDC Simulation	138
56-62	Kalman Estimator Performance	144-155
63-66	Manual Heading Mode Simulation Results	156-159
67-73	TACAN Mode Simulation Results	160-167
74-92	ILS Localizer Simulation Results	168-186
93-95	ILS Glideslope Simulation Results	187-189
96-99	Flight Test Data	192-195
100-108	Flight Test Data	199-207
109	Wait Loop and Off Mode Control	212
110	STM Mode Control	213
111	Manual Heading Mode Control	214
112	TACAN Manual Heading to Capture	215
113	TACAN Capture Computation	216
114	TACAN Capture to Track	217
115	TACAN Track to Overstation	217
116	TACAN Overstation to Capture	218
117	TACAN Mode Control	219
118	Lat. Common	220
119	DME Range Smoothing Algorithm	221
120	ILS Localizer Manual Heading to Capture	222
121	ILS Localizer Capture to Track	223
122	ILS Localizer Mode Control	224
123	GS Idle to Pre-capture	225
124	GS Pre-capture to Capture	226
125	GS Capture to Track	226
126	Glideslope Mode Control	227
127	Glideslope Common	228
128	TACAN Mode Timing (All Models Except ILS)	229
129	ILS Mode Timing	230
130	Mode Control	231
131	971D-1 Digital Flight Director Test Set	314
132	971S-1 CAPS Test Bench	315

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	DFDC Input Conversion Data	9
2	Digitizing Error Prediction	12
3	DFDC Output Signals	14
4	Output Conversion Error Predictions	15
5	I/O Addresses	19
6	I/O Discrete Bit Positions	20
7	Summary Characteristics of CAPS-2	24
8	ALU Element Operation Table	34
9	DFDC Performance Results T-39 Flight Tests	197
10	DFDC Performance Results	208

LIST OF SYMBOLS AND ABBREVIATIONS

ATE	- Along track error
ATGSE	- Angle to glideslope, predicted
ATGSN	- Angle to glideslope, best estimate
ATSE	- Along track slant wind error
ATSW	- Along track slant wind
CTE	- Cross track error
CTW	- Cross track wind
CWH	- Cosine of wind
G	- Gravity
GAME	- Gamma predicted
GAMH	- Gamma best estimate
GDF	- Glideslope beam rate, filtered
GDH	- Glideslope beam rate, best estimate
GH	- Glideslope beam deviation, best estimate
GSCORT	- Glideslope capture or track
HPTR	- Horizontal pointer
JO	- Sample time, seconds
KY	- Scaling term
LC	- Cross track wind filter coefficient
LATSW	- Later axis switch
OH	- Wind angle, best estimate
PHI	- Bank angle
PHIC	- Bank angle command
PHICL	- Bank angle command, capture
PHICT	- Bank angle command, track
PSI	- Heading
ODB	- Lateral beam rate, best estimate

LIST OF SYMBOLS AND ABBREVIATIONS (Continued)

QH	- Lateral beam deviation, best estimate
QHO	- Previous value of QH
RA	- Measured range
RE	- Predicted range
RH	- Estimated range
RHO	- Previous value of RH
RPN	- Radius of curvature
SA	- Slant range, measured
SE	- Slant range, predicted
SH	- Slant range, best estimate
SHO	- Previous value of SH
T	- Sample time, hours
TC	- Pitch command, capture
TDC	- Pitch rate command, capture
TDCA	- Pitch command, track
TDM	- Pitch rate, measured
TF1	- Pitch, filtered
TF15	- Pitch, filtered
TF15C	- Pitch rate
TM	- Pitch angle, measured
UO	- Airspeed
UF2	- UO acceleration
VHG	- Ground velocity, best estimate
VHTN	- Velocity tangent to RPN, best estimate
VPTR	- Vertical Pointer
WH	- Wind magnitude, best estimate
XSIC	- GAMH, filtered
YDT	- Cross track rate, \dot{y}

LIST OF SYMBOLS AND ABBREVIATIONS (Continued)

$C(k)$	= Time varying gain in the linear process underlying the Kalman filter
$E[]$	= Expected value of []
f_k	= Forcing function vector in dynamical model
G	= Wind vector magnitude (glideslope)
g	= Wind vector angle (glideslope)
H_k	= Measurement distribution matrix
J	= Sampling period of the decision process
K_k	= Kalman gain matrix
P_k	= Matrix in Kalman filter computation (P_0 = covariance matrix for \underline{x}_k)
R	= Localizer - distance from ground transmitter
r	= Radius of turn (localizer)
S	= Glideslope - distance from ground transmitter
U_0	= Airspeed
V_g	= Ground speed
V_T	= Velocity tangential to the vertical plane trajectory
W	= Wind vector magnitude (localizer)
X	= Rectilinear coordinate aligned with beam center line
\underline{x}_k	= Dynamical model state vector
$x_1(k)$	= $(R\theta)_k$ - Dummy variable to simplify notation in Kalman filter discussion
$x_2(k)$	= $\frac{d}{dt} (R\theta)_k$ - Dummy variable to simplify notation in Kalman filter discussion
$x_3(k)$	= Dummy variable to "whiten" noise for Kalman filter
Y	= Rectilinear coordinate perpendicular to beam center line
z_k	= Measurement (scalar)
γ	= Inclination of glideslope
δ	= Angle between aircraft center line and glideslope beam center line
ϵ	= Position apparent error vector magnitude (localizer)

LIST OF SYMBOLS AND ABBREVIATIONS (Concluded)

ζ	= Glideslope - beam angular deviation
η	= Radius of turn (glideslope)
θ	= Localizer - beam angular deviation
$v(k)$	= Measurement noise
ξ	= Position apparent error vector angle (localizer)
$\rho(k)$	= "Whitened" beam noise
$\rho'(k)$	= $(R\theta)$ beam noise
\underline{p}_k	= Noise vector in dynamical model
Σ	= Position apparent error vector magnitude (glideslope)
σ	= Position apparent error vector angle (glideslope)
ϕ	= Bank command
$\Phi_{k,k-1}$	= Transition matrix in dynamical model
ψ	= Course datum - angle between aircraft center line and localizer beam center line
Ω	= Pitch attitude
$\dot{\Omega}$	= Pitch rate command
ω	= Wind vector angle (localizer)

SUPERSCRIPTS

$(\vec{})$	= Vector quantity
$(\hat{})$	= Estimated quantity
$()'$	= Predicted quantity
$(\dot{})$	= Time derivative
$(\bar{})$	= Expected value (same as $E[\]$)
T	= Matrix transpose

SUBSCRIPTS

$()_k$	= Time index ($t_k = kJ$)
()	= Vector or $n \times 1$ matrix

SUMMARY

The objective of this program was to develop a Digital Flight Director Computer capable of adapting to different aircraft types and varying geometric conditions. This required the implementation of a multi-stage decision process and an error correction system. A predictive corrective technique is utilized to compute aircraft velocity, distance to station and mean wind allowing for compensation of these terms during beam capture. To aid in the estimation process, a Kalman filter was developed. The capture computations direct the aircraft along the arc of a circle tangential to the aircraft flight path and the center line of the radio beam.

In order to demonstrate the capability of the control to adapt to widely differing aircraft, a simulation effort was performed using four aircraft models; a jet transport, a propeller driven aircraft, a business jet, and a jet fighter. The results of these simulations verified the effectiveness of the control law for all four aircraft.

Implementation of the above techniques required the development of a high speed general purpose digital computer. The interface between the computer and existing aircraft attitude and navigation systems was accomplished by developing digital to analog and analog to digital conversion techniques tailored to the aircraft signals.

Two prototype 562A-14 Digital Flight Director computers and one 377B-1 Airspeed Controller were designed and fabricated for a flight test program.

Airworthiness of the 562A-14 was confirmed according to the Equipment Test Plan during environmental tests at Collins. The unit was installed in a T-39 aircraft at Randolph Air Force Base and operated successfully during flight tests in all modes: OFF, Manual Heading, TACAN and ILS.

1.0 INTRODUCTION

This document has been prepared as a final report with the intent to communicate the complete design of the 562A-14 Digital Flight Director Computer, developed under Air Force Contract F33657-73-C-0120.

The performance of present analog flight director computers varies with aircraft dynamics, geometry, and airspeed. These computers can only be adjusted to give optimum performance over a relatively narrow range of conditions, and are not capable of being directly interchanged between different types of aircraft without changes in the control laws. Once adjusted for the aircraft, performance is optimized only for a finite range of airspeeds and geometric conditions (geometry of the ground beam, and position of the aircraft relative to the beam). Their basic inability to derive ground speed and distance to station results in a sluggish response at long distances and excessively fast underdamped response at short distances from the ground station.

The objective of this program was to develop a universal flight director computer capable of providing optimum performance for any aircraft under all flight conditions, without requiring any changes in the computer. Additionally the flight director was to be independent of the aircraft equations and would not require storage of the aircraft dynamics. Performance requirements and unit specifications were set forth in the Statement of Work for the Digital Flight Director Computer.

This document is a comprehensive report on the total program which included formulation of a universal flight director control law, software and hardware prototype development, and simulation and flight tests performed to verify the system. Section I introduces program activities. Section II contains a detailed description of the prototype hardware. Section III is an analytical discussion of the Kalman Filter and of the Manual Heading, TACAN, and ILS control laws. Section IV provides a detailed presentation of the software implemented. Simulation study results are presented in Section V. Section VI contains flight test results.

Flow charts of the Digital Flight Computer Software are presented in Appendix A. Definitions of the terms used within the software programs are contained in Appendix B. Program listings of DFDC software in floating point Fortran, scaled fraction Fortran, and AED are presented in Appendices C, D, and E.

SECTION II

2.0 HARDWARE IMPLEMENTATION

2.1 General Characteristics

The 562A-14 Digital Flight Director Computer (DFDC), Collins Part No. 622-1824-001, shown in Figure 1 contains a general purpose digital computer, an aircraft systems interface unit and a rear mounted power converter. The digital computer and interface unit are constructed on eleven multilayer printed circuit cards. Flight control programs are stored internally in programmable read only memories.

The unit is contained in an ARINC one-half ATR short case. A 100 pin aircraft interface connector, a 100 pin test connector, an elapsed time indicator and a press to test push button and failure annunciator are located on the front panel.

The 377B-1 Airspeed Controller (ASC), Collins Part No. 622-1825-001, shown in Figure 2 was developed to provide mode selection and presettable airspeed inputs to the DFDC.

2.2 Interfacing Systems

2.2.1 Flight Director Inputs

An interface is effected with the following aircraft systems to provide the DFDC with the input signals listed below:

Aircraft Attitude System

- Roll (degrees)
- Pitch (degrees)

Horizontal Situation Indicator

- Manual Heading Error (degrees)
- Course Error (degrees)

Glideslope Receiver

- Glideslope Deviation (microamps)

Localizer Receiver

- Localizer Deviation (microamps)

TACAN Receiver

- TACAN Deviation (degrees)

Radar Altimeter

- Radar Altitude (feet)

DME Receiver

- Range (miles)

Air Data Computer

- True Airspeed (Knots)

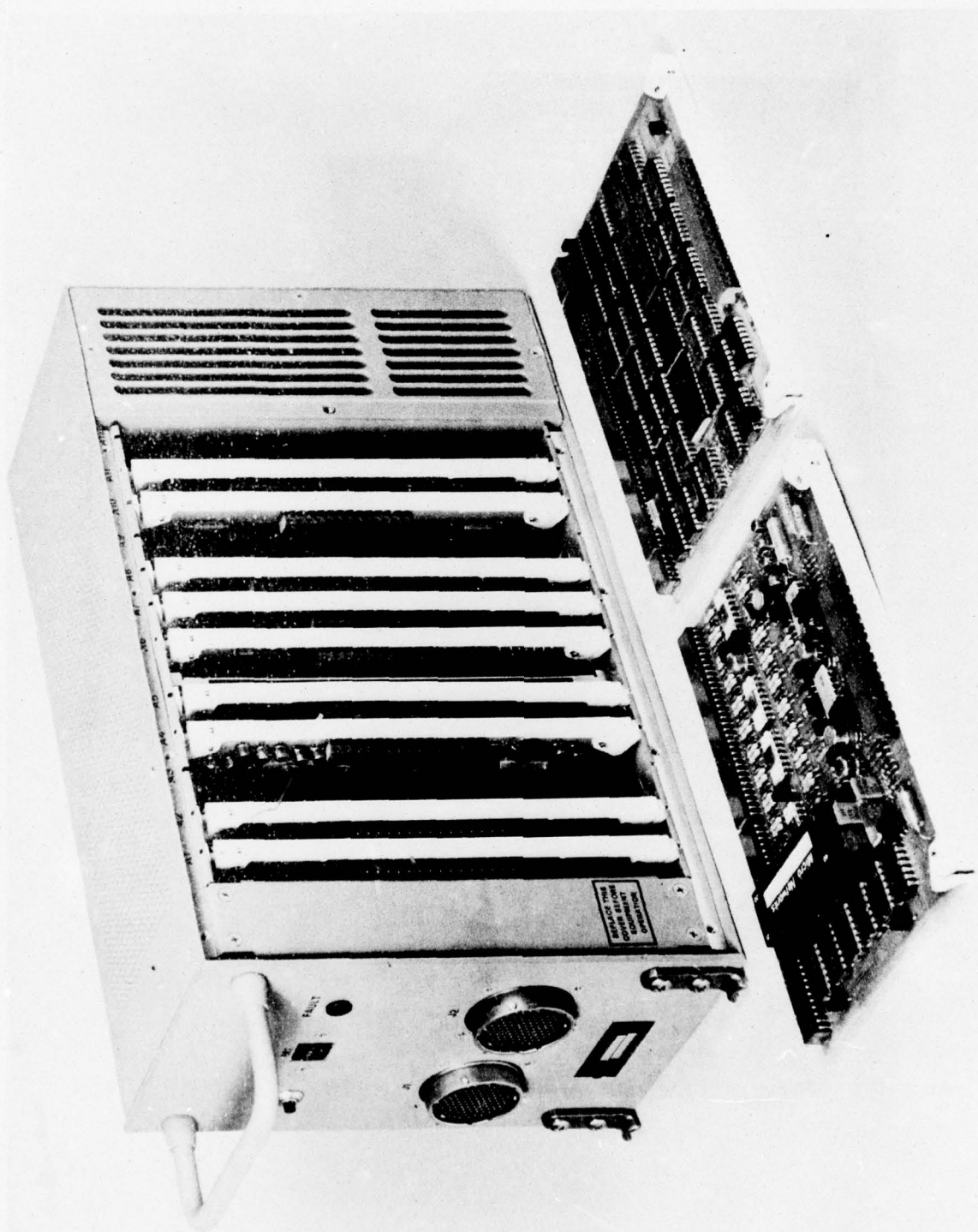


Figure 1 562A-14 Digital Flight Director Computer

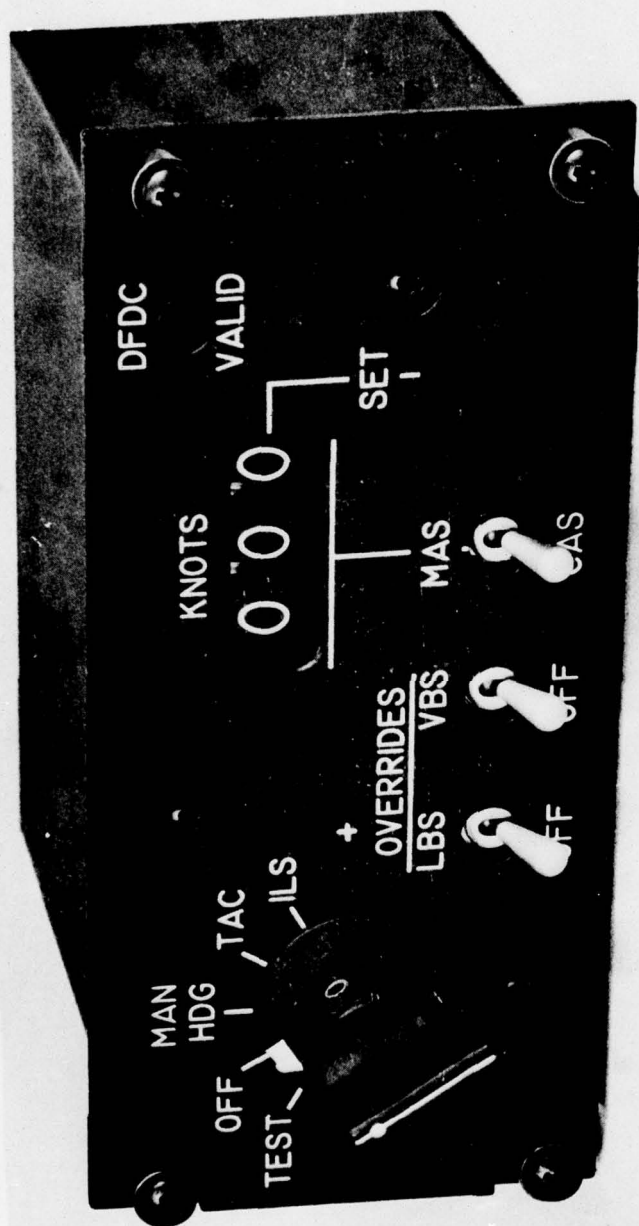


Figure 2. 377B-1 Airspeed Controller

Airspeed Controller

- Mode
- Airspeed (knots)

2.2.2 Flight Director Outputs

The DFDC supplies the attitude Director Indicator and Horizontal Situation Indicator with the output signals listed:

Attitude Director Indicator

- Vertical Pointer Position
- Vertical Pointer Flag
- Horizontal Pointer Position
- Horizontal Pointer Flag
- Glideslope Deviation
- Glideslope Flag
- Glideslope Engage
- Flight Director Valid

Horizontal Situation Indicator

- Lateral Deviation
- Lateral Deviation Flag

23 Block Diagram

The DFDC Block Diagram is shown in Figure 2.3. Each circuit function is identified with its card assembly. Major functions located on the five card assemblies of the aircraft interface unit are as follows:

- A1 DME UNITS CONVERTER
(SYNCHRO TO DC SIN, COS DEMODULATORS)
- A2 PITCH, ROLL CONVERTERS
(SYNCHRO TO DC SIN, COS DEMODULATORS)
- ILS, TACAN RECEIVERS
(MODULATED DC TO DC CONVERTERS)
- AIRSPEED, RADAR ALTITUDE DC RECEIVERS
- LOW LEVEL FLAG RECEIVERS
OUTPUT BUFFERS (GS, LATERAL DEVIATION)
- A3 COURSE & HEADING ERROR CONVERTERS
(CONTROL TRANSFORMER TO DC ANGLE DEMODULATORS)
- ANALOG MULTIPLEXER
- SAMPLE AND HOLD
- ANALOG TO DIGITAL CONVERTER

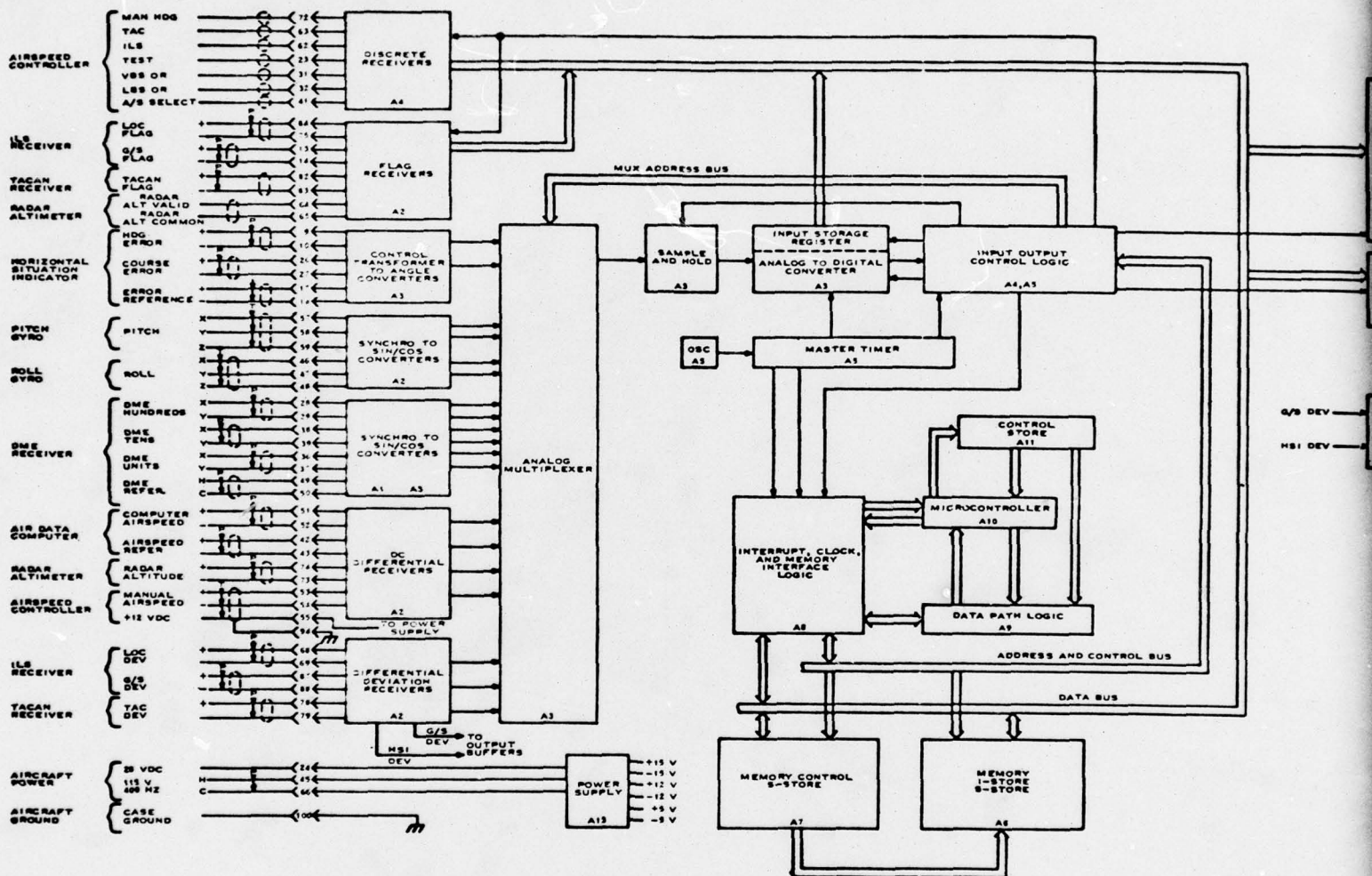
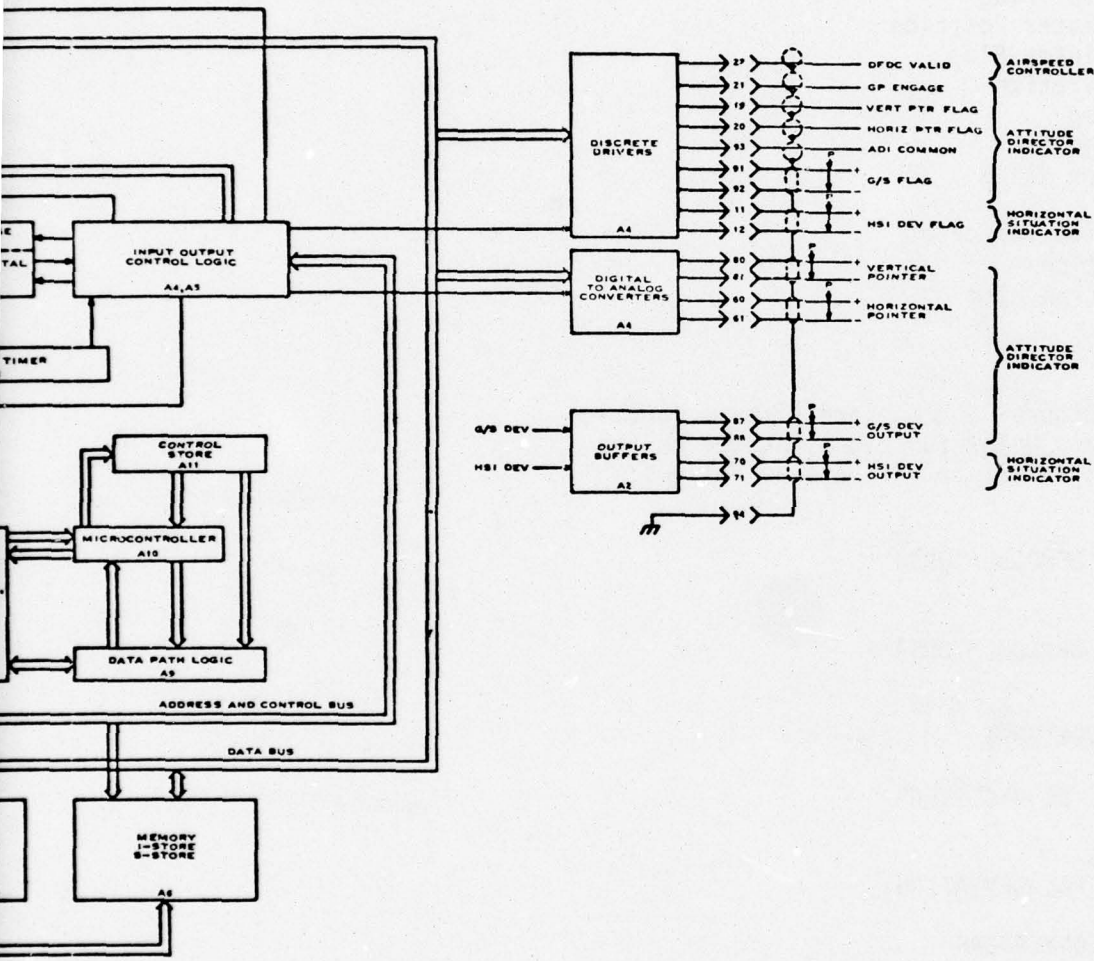


Figure 3. DFDC Block Diagram



DFDC Block Diagram

2

INPUT STORAGE REGISTER

A4 DISCRETE RECEIVERS

OUTPUT CONTROL LOGIC

DISCRETE DRIVERS

DIGITAL TO ANALOG CONVERTERS

A5 MASTER OSCILLATOR AND TIMER

I/O CONTROL LOGIC

The aircraft interface unit provides the link between the digital processor and the aircraft analog systems. Analog input signals are filtered, demodulated and scaled by input converters and supplied to an analog multiplexer. The output of the multiplexer is connected to an analog to digital converter through a sample and hold circuit. Twelve bits of data from the analog to digital converter are placed in the input storage register and enabled onto the data bus on processor request.

All data transactions between the aircraft interface unit and the processor are conducted on the transfer bus which consists of a 16 bit data bus, a 12 bit address bus and 6 control lines.

The processor selects input data by forwarding a five bit code via the address bus to the I/O control logic which selects the appropriate multiplex channel and initiates A to D operation. During the time that analog data is being digitized the processor proceeds with other computations. Once the data is ready the I/O control logic sends an interrupt to the processor. The processor then requests the data in the input storage register. Data is then placed on the transfer bus. Discrete inputs and mode control signals are converted to TTL logic levels and placed on the transfer bus as requested by the processor.

After the completion of processor computations, output data is supplied to the vertical and horizontal pointer D to A converters and discrete drivers via the transfer bus. New data is sent to these converters once every 100 milliseconds.

The computer implemented in the DFDC is a member of the Collins adaptive processing system (CAPS) family of computers. This processor, designated CAPS-2, is a general purpose stack oriented digital computer. It is utilized to perform flight control computations, mode switching, and self monitoring. The basic unit contains a 16 bit parallel execution unit referred to as the central processing unit (CPU), a 1024 word by 16-bits-per-word high speed mos read-write-memory and a 4096 word by 8-bits-per-word high speed bipolar programmable read-only-memory that contains program instructions. Machine architecture is consistent with compiler techniques to allow programs written in high-level language to be compiled into efficient machine language programs.

Major functions located on the six card assemblies of the CAPS-2 processor are as follows:

- A6 I STORE (INSTRUCTION MEMORY)
S STORE (STACK MEMORY)
- A7 MEMORY CONTROL
S STORE
- A8 INTERRUPT, CLOCK MEMORY INTERFACE LOGIC
- A9 DATA PATH LOGIC (CPU)
- A10 MICROCONTROLLER
- A11 CONTROL STORE

2.4 Aircraft Systems Interface

2.4.1 Input Parameters

The conversion and scaling of analog input parameters to a digital format compatible with the CAPS-2 processor is accomplished by the aircraft interface unit. Table 1 describes signal format and scaling for each input received.

2.4.2 Input Circuitry

Dual micropower operational amplifiers were selected for input receiving, scaling, and demodulation functions to minimize power dissipation and board space.

2.4.2.1 Synchro Demodulators (CARD A2)

Pitch and Roll synchro inputs are converted to sin and cos and demodulated on the A2 card assembly. In order to improve the accuracy of these dc outputs the standard 26V reference has been replaced as a demodulator phase control signal by the 400 cycle cosine term derived from the synchro inputs. Since the typical maximum value of roll and pitch inputs is 45 degrees and 20 degrees respectively, the cosine signals will be maintained at not less than 7 volts, and can be utilized as a phase reference for both sin and cos terms.

2.4.2.2 Radio Receivers (CARD A2)

The 90 to 150 Hz modulated Localizer, Glideslope, and TACAN signals are received differentially, amplified, filtered, and outputted to the analog multiplexer. A two stage second order filter is utilized to provide 60 db of attenuation at 100 Hz. and to minimize delay of the input signal. Phase delay of the Localizer Glideslope and TACAN receivers is 0.5 degrees at 0.1 radian.

The differential receivers used for Radar Altitude and Airspeed provide 20 db of 400 cycle attenuation and result in a phase delay of 8 degrees at 1 radian.

TABLE 1. DFDC INPUT CONVERSION DATA

Signal Type	Input Scale	Input Max. Value Expected	I/O Output Form	Sign Bit Weight	I/O LSB Weight	Rcvr Gain	Input Impedance
Pitch	Synchro	+20°	Sinθ Cosθ	(1)		Demod	X-Z, Y-Z 47.4K X-Y 23.7K
Roll	Synchro	+45°	Sinθ Cosθ	(1)		Demod	X-Z, Y-Z 47.4K X-Y 23.7K
DWE Range	Synchro (2)	300 NM	Sin Cos	(1)		Demod	X-Z, Y-Z 47.4K X-Y 23.7K
Hdg Error	Control Transformer	+90°	ψ	-90°	.04939°	Demod	68K
CRS Error	Control Transformer	+90°	ψ	-90°	.04939°	Demod	6.8K
LOC Dev	Mod. DC	+300ua +4°	Angle	-5.625° -421.875ua	.002747° 0.2060ua	23.704	1K
TAC Dev	Mod. DC	+150ua +10°	Angle	-11.25° -168.75ua	.00549° .08240ua	59.259	1K
G/S Dev	Mod. DC	+150ua +0.7°	Angle	-1.40625° -301.339ua	.000687° 0.14714ua	33.185	1K
Airspeed	DC 0 → 12V	12VDC 840 kts	Knots	-840 kts -12V	0.4102 kts .00586V	0.8333	240K
Radar Alt	DC 0 → -40V	-40VDC 1000 feet	Naut. Mi.	-6070.1155 ft. -242.805V	2.9639 ft. 0.11856V	0.0412	127K

(1) Synchro inputs are normalized by the software.
 Nom. value for Sin = 1 or Cos = 1 is 9 volts - 3463g

JD 6/5/74

2.4.2.3 Heading & Course Error Demodulators (CARD A3)

Course and Heading Error control transformer outputs from the HSI are demodulated by utilizing the 26 V reference from the HSI heading synchro. In order to provide constant demodulator output with a variable reference voltage the 26V reference is compared with the demodulated sin and cos input signals. The filtered result of this comparison is a dc output directly proportional to error angle.

2.4.2.4 Flag Receiver (CARD A4)

Glideslope, Localizer, and TACAN flag signals are accepted by differential receivers with 1K input impedance. Noise immunity is provided by schmitt triggers at the receiver outputs. Trip points and hysteresis are illustrated in Figure 4 .

2.4.2.5 Analog Multiplexer (CARD A3)

The sixteen scaled dc voltages from the synchro demodulators and differential receivers are multiplexed through complimentary MOS analog switches to the summing junction of an operational amplifier. This arrangement permits zero voltage current switching to minimize linearity and cross-talk errors.

2.4.2.6 A to D Conversion (CARD A3)

The Analog Multiplexer output is connected directly to a sample and hold function to insure that the A to D converter input voltage will remain constant during the conversion process. A successive approximation analog to digital converter was chosen for this application to provide conversion times of less than 200 usec. The total time required for signal conversions is 3.2 msec. The 12 bit A to D provides 5 mv of resolution and 10 mv of accuracy over the -55 to +125 C temperature range. Input signal range is -10 to +10 volts.

2.4.2.7 Digitizing Accuracy

The worst case errors introduced by DFDC scaling and digitizing are listed in Table 2 . Predicted errors were obtained by determining the maximum offset and gain error for each circuit in a signal path, and assuming all errors to be additive. The root sum squared of these errors is entered in Table 2 along with results obtained from laboratory breadboard operating over the -55 C to +125 C temperature range. Laboratory tests were conducted to verify the critical offset errors of the complex synchro demodulator circuits utilized for ROLL, PITCH, DME, and CRS and HDG Error inputs. Non-military parts were utilized for these tests and final configuration DFDC units are expected to provide improved accuracy. Since final configuration 0.1% resistors were not available at the time the tests were performed, 1% resistors were utilized and the observed gain errors were much greater than predicted.

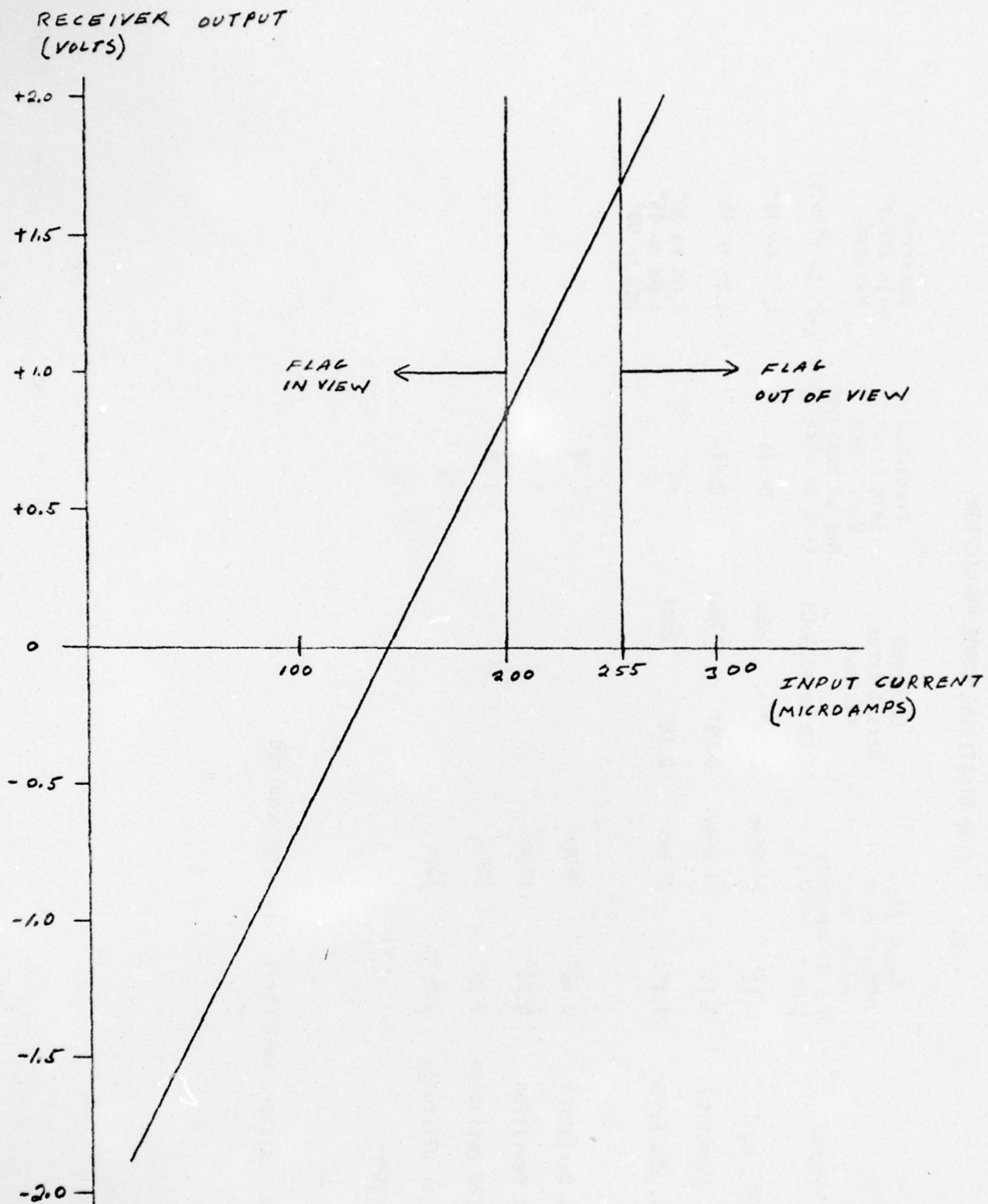


Figure 4. Flag Receiver Action

TABLE 2. DFDC DIGITIZING ERROR PREDICTIONS

Signal	Predicted Offset Error Worst Case Rss Of Maximums (-55 to +125°C)	Observed Offset Error Maximum (-55 to +125°C)	Predicted Gain Error Worst Case Rss of Maximums (-55 to +125°C)	Observed Gain Error* Maximum (-55 to +125°C)
Pitch, Roll	0.13°	14.85mv	0.09°	11mv
DME (Synchro)	0.13°	14.85mv	0.09°	11mv
CRS, HDG Error	0.3°	33.6mv	0.18°	20mv
G/S Deviation	0.02°	141mv		1.3%
Loc Deviation	0.06°	103mv		1.3%
TACAN Deviation	0.27°	245mv		1.3%
Radar Altitude	7.9 ft	13mv		1%
Airspeed	1.17 kts	14mv		1%

*1% resistors used where 0.1% were required

JD 8/28

2.4.3 Output Parameters

The DFDC outputs analog pitch and bank steering commands, glide-slope deviation, and flag control logic to the Attitude Director Indicator. It provides the Horizontal Situation Indicator with lateral beam deviation and radio validity logic. Characteristics of DFDC outputs are listed in Table 3.

2.4.4 Output Circuitry

The aircraft interface unit converts digital data received from the processor at a refresh rate of 10HZ, to analog outputs compatible with the ADI and HSI. Two 12 bit A to D converters are provided for pitch and roll steering commands. The resolution and refresh rate of the A to D converters insures smooth, jitter free outputs. Flag and validity outputs are converted from logic levels to indicator compatible inputs by discrete transistor circuitry.

2.4.4.1 Output Control Logic (CARD A4)

Output data received from the processor is stored in three registers, implemented with complimentary mos devices. Four parallel in serial out shift registers provide storage for the Horizontal and Vertical Pointer data. Each time the registers are updated new data is transferred serially to the D to A converters, under the direction of the output control logic. Independent control functions for the A to D converters allow processor refreshing of either register during serial data transfer to the converters. The control logic responds to the output "data accept" signal from the I/O control and determines the processor selected output signal by interrogating the least significant three bits of the Address Bus. Discrete control bits are received from the Data Bus and stored in an eight bit complimentary mos register. The discrete output drivers are designed to source 300 ua to loads from 400 to 1200 ohms and will drive either one or two external flags.

2.4.4.2 Output Accuracies

Table 4 describes the predicted worst case error for DFDC output conversion circuitry as compared with specified accuracies within MIL C 83014A.

2.4.5 I/O Control Logic (CARD A5)

Digital data transfer, between the I/O conversion circuitry and the CAPS-2 processor, is accomplished by utilizing the processor data and address buses and the control signals shown in Figure 5. The I/O control logic monitors address bus and control signals to determine when I/O action is requested and the operation required as illustrated in Figure 6.

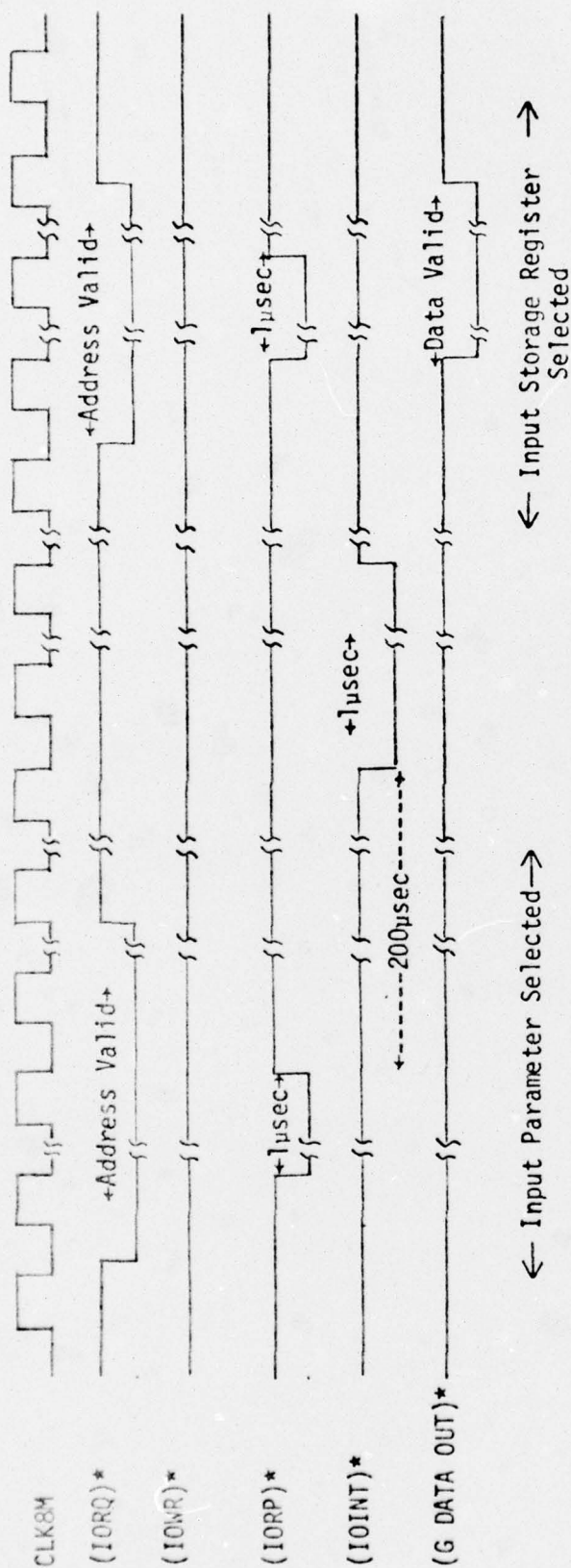
TABLE 3. DFDC OUTPUT SIGNALS

Output Signal	Signal Type	Output Scale	Range	Sign Bit Weight	LSB Weight	Output Impedance	Control Signal
Horizontal & Vertical Pointers	Analog	13.03°/ma	-5 to +5 rad	-5 radians	.014 radians	140Ω	DB08
			-2.2 to +2.2 ma	-2.2 ma	1.07 ua		DB09
			+12ma (out of view)				
G/S Deviation	Analog	214.29ua/Deg	+150ua ±0.7°			10Ω	
Lateral Deviation	Analog	15ua/Deg	+150ua, ±10°			10Ω	ILSF
1) TACAN mode							
2) ILS Mode		75ua/Deg	+300ua ±4°				ILS
3) OFF			750ua (out of view)				OFF
Vertical & Horizontal Ptr Flags	Discrete		0ua (in view) 300ua (out of view)			100Ω	DB 15 DB 14
Lateral Flag	Discrete		0ua (in view) 300ua (out of view)			100Ω	DB 13
G/S Flag	Discrete		0ua (in view) 300ua (out of view)			100Ω	DB 12
G/P Engage	Discrete		Open 160 ma sink			10Ω	(DB 11)*
DFDC Valid	Discrete		Open 160 ma sink			10Ω	Go/No Go (DB 10)*

TABLE 4. OUTPUT CONVERSION ERROR PREDICTIONS

Signal	Predicted Offset Error Worst Case (-55 to +125°C)	Predicted Gain Error Worst Case (-55 to +125°C)	Specified Maximum Error MIL-C-83014	Predicted Maximum AC Ripple (-55 to +125°C)	Specified Maximum AC Ripple MIL-C-83014
Horizontal Pointer	25mv eq. to 1%	0.5%	5%	15mv	150mv
Vertical Pointer	25mv eq. to 1%	0.5%	5%	15mv	150mv

INPUT TRANSACTIONS



OUTPUT TRANSACTIONS

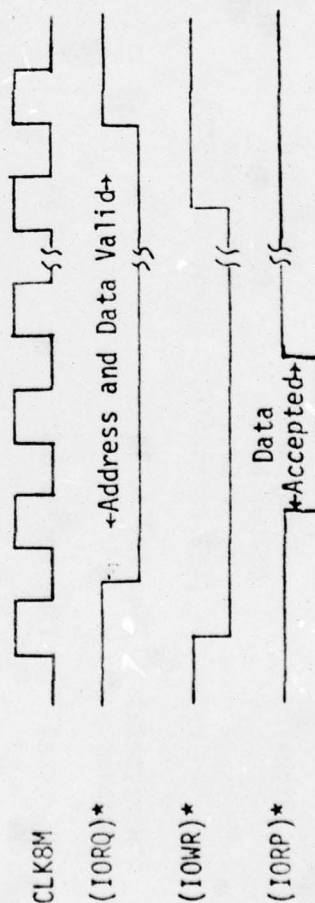


Figure 5. I/O Control Signals

$I/O \text{ Select} = IORQ \cdot [17700 \leq \text{Address} \leq 17737]$
 $I/O \text{ Select} = IORQ \cdot A_{12} \cdot A_{11} \cdot A_{10} \cdot A_9 \cdot A_8 \cdot A_7 \cdot A_6 \cdot (A_5)^*$
 $\text{Storage Register Select} = I/O \text{ Select} \cdot A_4 \cdot A_3 \cdot A_2 \cdot A_1 \cdot A_0$
 $\text{Data Output} = I/O \text{ Select} \cdot (IO \text{ WR})^*$
 $\text{Discrete Input Request} = I/O \text{ Select} \cdot A_4 \cdot A_3 \cdot A_2 \cdot A_1 \cdot (A_0)^*$
 $A \text{ to } D \text{ Conversion Request} = I/O \text{ Select} \cdot (IOWR)^* \cdot (DIR)^* \cdot (SRS)^*$

The following mnemonics apply:

IORQ - IO Request
 IOWR - IO Write
 DIR - Discrete Input Request
 SRS - Storage Register Select

Figure 6. I/O Control Logic

2.4.5.1 Data Input

Input parameters are selected by the five least significant bits of the address bus as illustrated in Table 5. Analog data is digitized and placed in the input storage register 200 microseconds after a processor request. An I/O generated interrupt is then sent to the processor which requests the storage register data. This operation permits the processor to perform other computations during input conversion. Discrete inputs are converted to TTL logic levels and placed on the data bus upon processor request. Location of each discrete input on the data bus is shown in Table 6.

2.4.5.2 Data Output

The I/O Control Logic accepts and stores processor output data and initiates the D to A conversion process. Since D to A control functions are independent, the processor is not required to wait for one conversion to complete before initiating another.

2.4.5.3 Oscillator and Master Timer (CARD A5)

A 16 mhz counter and synchronous timing chain generate clock signals for the DFDC. The following timer output signals are available at the A5 connector:

CLK8M - 8 MHZ clock

CLK4M - 4 MHZ clock

CLK2M - 2 MHZ clock

CLK1M - 1 MHZ clock

CLK.5M - 0.5 MHZ clock

CLK.25M - 0.25 MHZ clock

The master timer also provides the CAPS computer with two timer interrupts to initiate the main computation cycle and the glideslope computation. The interrupts are 1 usec in length and occur at 100 msec intervals. The time relationship between timer 1 and timer 2 may be adjusted by moving one jumper wire located on the A5 card assembly. The diagram of Figure 7 shows the timer relationships available.

The master timer utilizes low power schottky synchronous counters to provide divisions of 16, 14, 15, 15 and 16. An 8 mhz input to the timer is divided down to 9.921 hz. In order to minimize skew between the 8 mhz and 9.9 hz clocks the master timer is implemented with look ahead enabling gates which effectively eliminate the race conditions characteristic of long odd multiple dividing chains.

TABLE 5. I/O ADDRESSES

	<u>ADDRESS</u>	<u>PARAMETER</u>	<u>ADDRESS</u>	<u>PARAMETER</u>
	177 00	Vertical Pointer	177 32	Self Test Constant (+)
	177 01	Horizontal Pointer	177 33	RUSIN
	177 02	Output Discretes	177 34	Airspeed Reference
Output	177 03	Spare	177 35	RUCOS
Data	177 04	Spare	177 36	Discretes
	177 05	Spare	177 37	Input Storage Register
	177 06	Spare		
	177 07	Spare		
	177 10	Pitch Sin		
	177 11	Pitch Cos		
	177 12	Roll Sin		
	177 13	Roll Cos		
	177 14	Range T Sin		
	177 15	Range T Cos		
	177 16	Range H Sin		
	177 17	Range H Cos		
Input	177 20	Crs Error		
Data	177 21	Hdg Error		
	177 22	Loc Dev		
	177 23	Tac Dev		
	177 24	G/S Dev		
	177 25	Comp AS		
	177 26	Man AS		
	177 27	Radar Alt		
	177 30	Horiz Pointer		
	177 31	Vertical Pointer		

TABLE 6. I/O DISCRETE BIT POSITIONS

<u>Data Bus</u>	<u>Inputs</u>	<u>Outputs</u>
DB00	MAN HDG	
DB01	ILS	
DB02	TAC	
DB03	SELF TEST	
DB04	MAN AS	
DB05	VBS OR	
DB06	LBS OR	
DB07	TAC FLAG	
DB08	LOC FLAG	Horiz Ptr In View
DB09	G/S FLAG	Vert Ptr In View
DB10	RALT VALID	Processor No Go
DB11		(GP Engage)*
DB12		G/S Flag In View
DB13		Loc Flag In View
DB14		Horiz Ptr Flag In View
DB15		Vert Ptr Flag In View

Notes:

- 1) Data Bus signals in the logic 1 state imply that the flags are in view.
- 2) A selected mode will result in a logical 1 in its respective bus bit.
The OFF mode is defined as: $DB00 + DB01 + DB02 = 0$

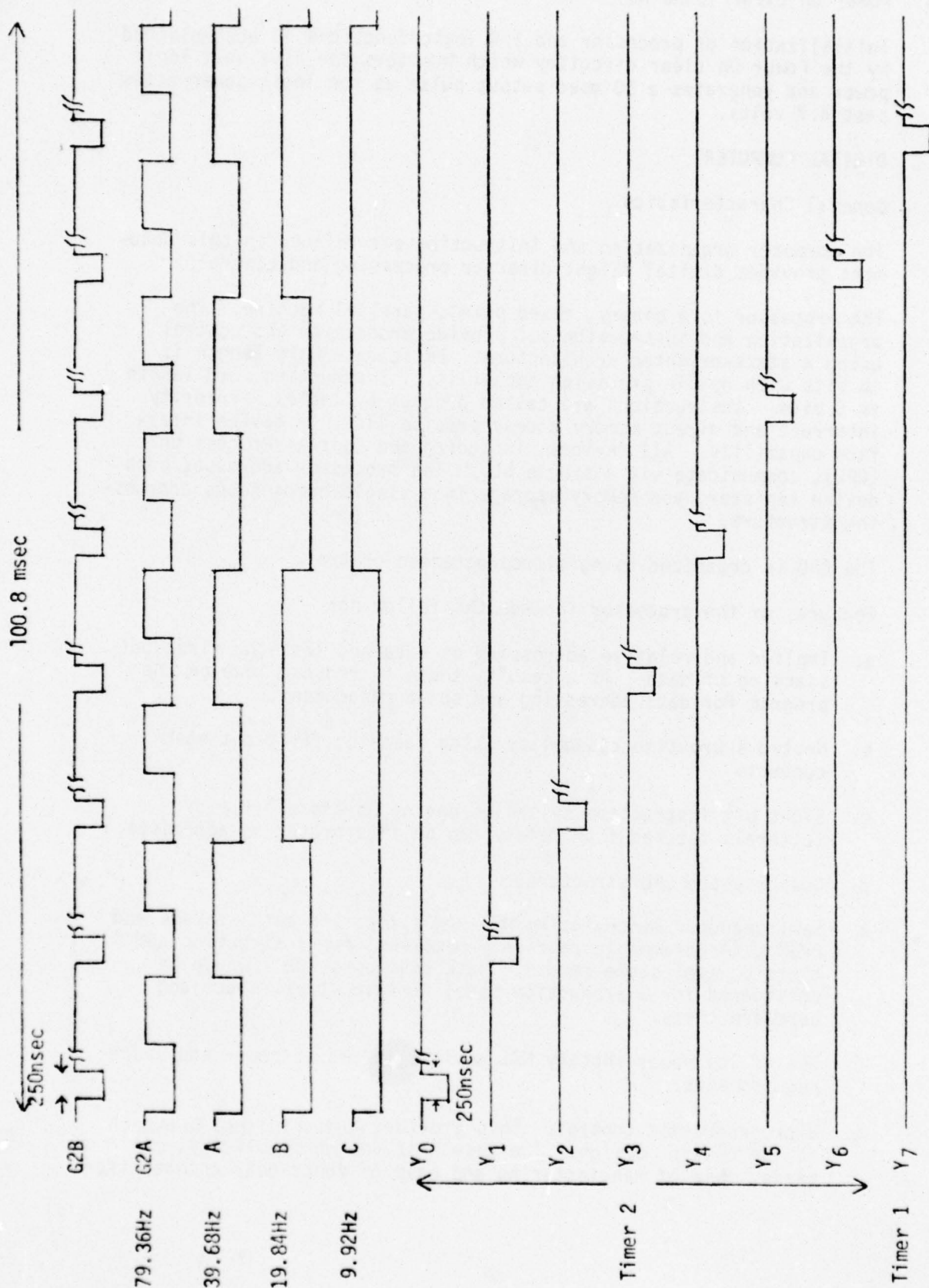


Figure 7. Timer Interrupts

2.4.5.4 Power On Clear (CARD A5)

Initialization of processor and I/O logic functions is accomplished by the Power On clear circuitry which monitors the five volt logic power and generates a 50 msec output pulse as the logic power rises past 4.7 volts.

2.5 DIGITAL COMPUTER

2.5.1 General Characteristics

The computer organization and instruction set defined in this document provides digital flight director processing and control.

The processor is a binary, fixed point, parallel machine. The organization and instruction set provide processing and control using a stack-oriented architecture. Basic data word length is 16 bits with double precision capability. Instruction word length is 8 bits. Instructions are called program syllables. Priority interrupt and direct memory access provide flexible device interface capability. All devices, including the Central Process Unit (CPU), communicate via a single bus. The processor addresses both device registers and memory storage in a single homogeneous addressing structure.

The CPU is organized using microprogrammed control.

Features of the processor include the following:

- a. Implied and relative addressing of data and last-in, first-out stacking of data. As a result, there is reduced load on the program for data addressing and space management.
- b. Nested subroutine capability using last-in, first-out stack concepts.
- c. Eight bit instruction syllables having no address fields. (Literals fetched from memory can be interpreted as addresses.)
- d. Bus oriented CPU structure.
- e. Semiconductor memory using MOS RAM's for main data storage and PROM's (Programmable read-only-memories) for instruction and micro control store memory. Mask developed ROM's would be considered for a production model to save power, space and hardware costs.
- f. Use of low power shottky MSI devices to reduce power and space requirements.
- g. Microprogrammed control. This provides a disciplined approach to the control design. The result is design simplicity, minimum parts, ease of manufacturing and ease of functional changes (for

example, modification of existing machine instructions or addition of new ones).

- h. The ability to supplement the machine language with microprogrammed subroutines which handle functions that cannot be accomplished in real time at the machine language level. Digital flight director subroutines include arithmetic operations and an input data interrupt handler.
- i. The architecture of the machine allows a program written in a high-level language to be easily compiled into a very efficient machine language program.

The basic characteristics of the machine are summarized in Table 7 .

TABLE 7. SUMMARY CHARACTERISTICS OF CAPS 2 (DIGITAL FLIGHT DIRECTOR)

Processor Organization:	LIFO (Last-in, first-out stack) concept.
Instruction Format:	8-bit syllables (zero address).
Data Type:	16-bit fixed point, fractional binary.
Instruction Count:	47
Execution Times:	
Add/Subtract:	3 microseconds.
Multiply:	22 microseconds.
Data Reference/Assignment:	3 microseconds typical.
Input/Output:	16-bit parallel data bus. Direct memory access. Program access to I/O devices. Priority interrupts.
Memory:	Semiconductor. 4096 8-bit instruction words. 1024 16-bit data words.
Power:	20 watts.
Size:	Less than 100 square inches.

2 5.2 STACK OPERATION

Within the computer is the stack, sometimes called a push-down store or LIFO (last-in, first-out) list. An operation implicitly specifies a storage location as a source or destination of data to or from the top of a list of contiguous storage words. Operands are referenced from the top of the list and the result operands normally replace the operands at the top of the list.

Conceptually, a push-down stack is a set of registers of which only one is externally accessible. Data is loaded or "pushed" into the stack through this one register. When new data is loaded, previously loaded data moves sequentially "down" from one register to the next in the set. All data is retrieved or "popped" in the reverse order from that in which it was loaded.

The actual implementation of the stack is in the main memory of the processor. A set of successive locations in the memory is used for the stack. Instead of data actually moving from location to location during a push or a pop, the address of the last entry in the stack is stored in a stack pointer. The pointer can be incremented or decremented and points to the location that is currently the top of the stack.

An example of stack usage is the evaluation of an arithmetic expression. Consider the following expression:

$$X = (A-B)/(C+D)$$

The expression can be formatted into "reverse Polish notation" (the commas may be omitted):

$$A,B,-C,D,+,/,X=$$

This notation eliminates parentheses because each mathematical operator refers not to the two operands between which it stands, but to the two operands preceding it - or, in the case of unary operators, to only the first preceding operand. (This is the translation that a compiler does to a high-level language prior to generating machine language.)

To evaluate an expression in Polish notation, the program contains a reference instruction (a "push") for each variable, a mathematical instruction (such as ADD) for each operator in the expression, and an assign instruction (a "pop") for the = sign.

Mathematical instructions operate on the top one or two items in the stack and replace them with the result. The assign instruction uses the address of the dependent variable X as the location in main memory for storing the result of the evaluation. This sequence is illustrated in Figure 8.

STACK EVALUATION

$$X = (A-B)/(C+D)$$

Polish notation: AB-CD+/X=

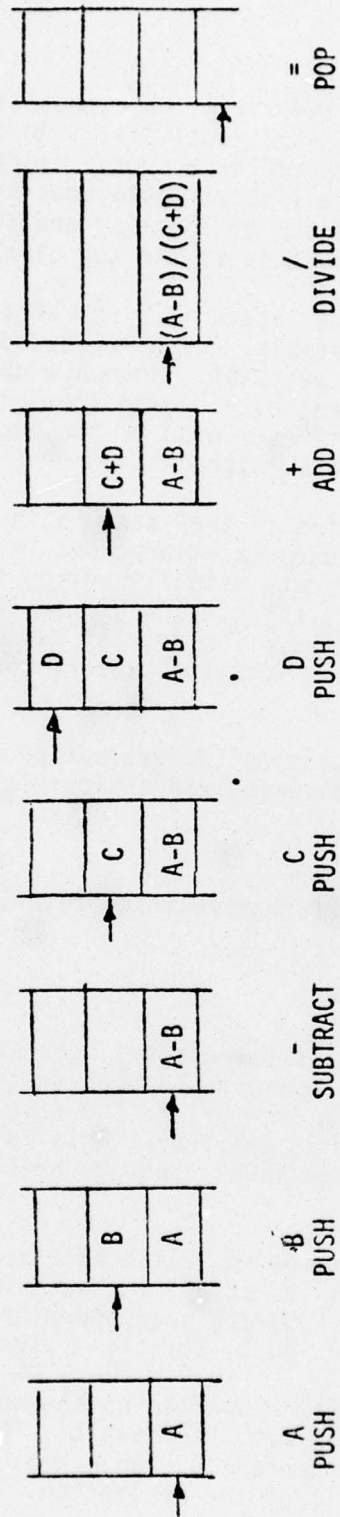


Figure 8. Stack Evaluation Example Operation

The computer implements the stack in a contiguous block of random access memory. (The random access memory is referred to as the S-memory.) Figure 9 presents this view in graphical form. The notation for the words within the accumulator stack are as follows:

- S0: The top word in the stack (the latest word entered and available to machine operations).
- S1: The second word in the stack, located in the next higher numbered memory location.
- S2: The third word in the stack, in a memory position next higher numbered from S1.

Theoretically, the number of words in the accumulator stack is unlimited. However, machine operations rarely require more than four words.

The stack function is controlled by pointers in scratch pad memory. These pointers are accessible only by the micro-control logic. They are as follows:

1. TOS (Top of Stack)

A 16-bit word in scratch pad memory (P-memory) contains the memory address (pointer) of the top word of the accumulator stack (i.e., the last word that was inserted into the stack and/or the first word to be removed). TOS can be incremented ('POP'), removing the top word of the stack, or can be decremented ('PUSH'), making room for an additional word in the stack. The top of the stack is redefined by TOS (the contents of the stack do not move).

The actual implementation of the digital flight director processor allows the top two words in the stack to be maintained in CPU registers. This reduces main memory accesses, increasing program execution speed. The algorithm for the implementation is discussed in the section describing the microcontroller.

2. LENV (Local Environment Pointer)

The empty condition of the accumulator stack occurs when the top of the stack pointer (TOS) equals the value of LENV, a 16-bit word in scratch pad containing the memory address of the first word (lowest numbered address) of the Local Environment. The Local Environment is a maximum of 32 16-bit words per subroutine located in memory which can be accessed by a class of instructions ("Local Environment Relative") and which has the relative address of the

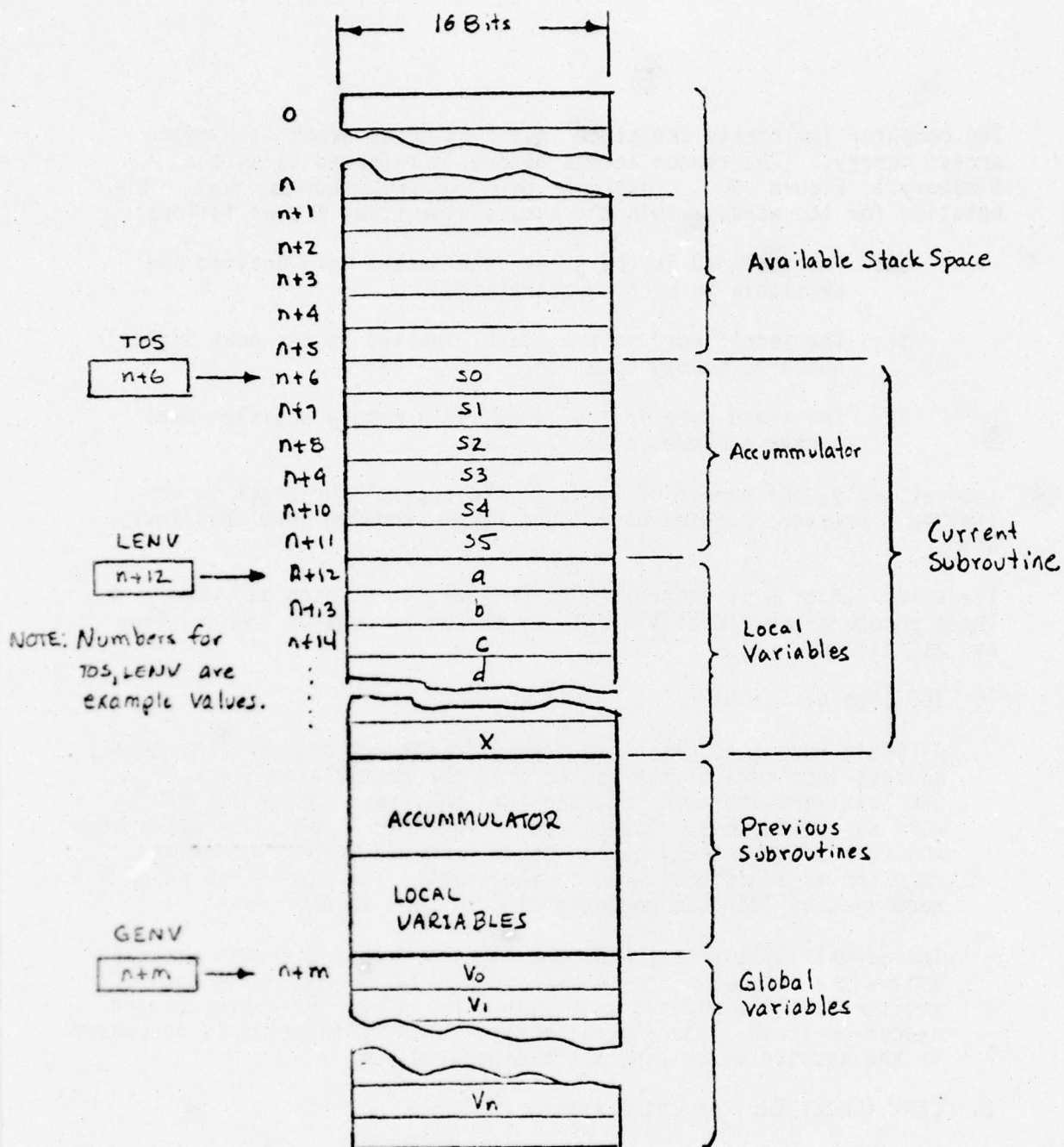


Figure 9. Stack Organization

word within the Local Environment as part of the instruction syllable. The accumulator stack is located "above" the Local Environment; i.e., the bottom word of the stack is in memory position, LENV minus one.

3. GENV (Global Environment Pointer)

A 16-bit register containing the starting address (lowest numbered) of the Global Environment. This area of memory contains those variables used by different user programs. Instructions, using GENV as a base, generate an absolute address for accessing any word of memory. Since there is only one user program in the digital flight director application, GENV has only one value, zero, and there is no need of a GENV register pointer.

4. SPCR (Program Syllable Counter)

SPCR is a 16-bit word in scratch pad memory capable of being loaded, incremented, and decremented in the ALU under micro-program control. The least significant 12 bits point to the syllable address (8-bit syllables) to be executed.

A characteristic of SPCR, common to all instructions, is that SPCR (during the execution of an instruction) is pointing to the address of the next syllable to be executed in a linear string of syllables. SPCR can, of course, be modified by transfer instructions or by Conditionals occurring within the body of an instruction.

The digital flight director processor organization is shown in Figure 10 . Basically, the processor consists of four functional elements as follows:

1. Data path logic:

The data path logic contains the mechanisms for data manipulation in the machine. The logic includes an ALU, a scratch pad memory, several registers, and main memory interface logic.

The data is manipulated and main memory is accessed under control of the microcontroller. The scratch-pad memory contains pointers that implement the program counter and space management functions of the machine.

2. Microcontroller:

The microcontroller provides the logical control of the data path logic. The control store memory (built from read-only-memory elements) contains a programmed-structured sequence of commands which are translated into hardware control signals.

Control store address and branch logic contains the necessary decision logic to allow execution of machine instructions fetched from the instruction memory. The decision logic can change the state of the machine as a result of external interrupts or instructions.

3. Main Memory:

This functional element provides storage for the macro-program and the random access space required for the execution of the program.

4. Interrupt Logic:

This logic allows external devices to change the state of the machine by alerting the microcontroller of some condition. Priority is assigned to each kind of interrupt. Interrupts can also be generated internally under micro programmed control.

A fifth function is that of direct memory access (DMA). This function is distributed among the various devices attached to the memory bus and allows each device (including the central processing unit) to contend for use of the memory bus. All devices communicate with the main memory via the memory bus.

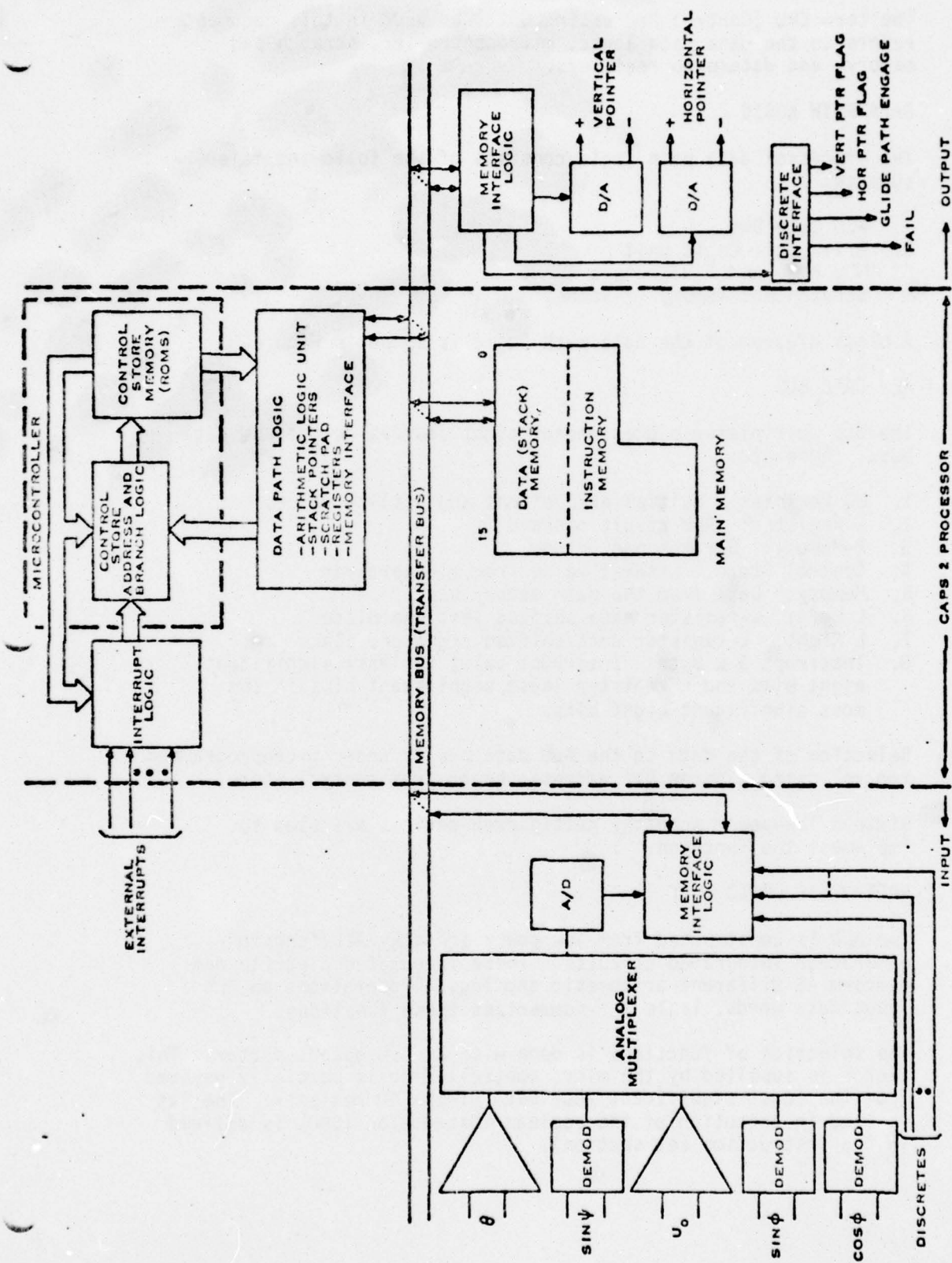


Figure 10. Processor Organization

The term CPU (Central Processing Unit) as used in this document refers to the data path logic, microcontroller, scratch pad memory, and data path registers.

2.5.3.1 DATA PATH LOGIC

The processor data path logic consists of the following major subsets:

1. ALU Data Bus
2. Arithmetic Logic Unit
3. MQ, B, L and A Registers
4. Scratch Pad Memory (P-Store)

A block diagram of the data path logic is shown in Figure 11 .

2.5.3.1.1 ALU DATA BUS

The bus multiplexer places one of eight sources on the ALU data bus. These are:

1. MQ Register: Multiplier/Quotient and utility values
2. L Register: ALU result operand
3. P-Memory: Scratch pad values
4. Control Store: Literal value from microprogram
5. Memory: Data from the main memory bus
6. L Left: L-register data shifted left one place
7. L Right: L-register data shifted right one place
8. Interrupt & L Byte: Interrupt value in least significant eight bits and L register least significant bits in the most significant eight bits.

Selection of the data to the ALU data bus is under microprogrammed control using a three bit address vector from control store.

- Sixteen low-power schottky multiplexer devices are used to implement the function.

2.5.3.1.2 ARITHMETIC LOGIC UNIT

The ALU is constructed from low power schottky ALU/function generator integrated circuits. These integrated circuits can perform 48 different arithmetic and logical operations on two input data words. Table 8 summarizes these functions.

The selection of functions is done with an ALU opcode vector. This vector is supplied by the micro controller or is partially derived from the least significant four bits of the MQ register. The lat is used in execution of the boolean instruction (BOOL is defined in the instruction set section).

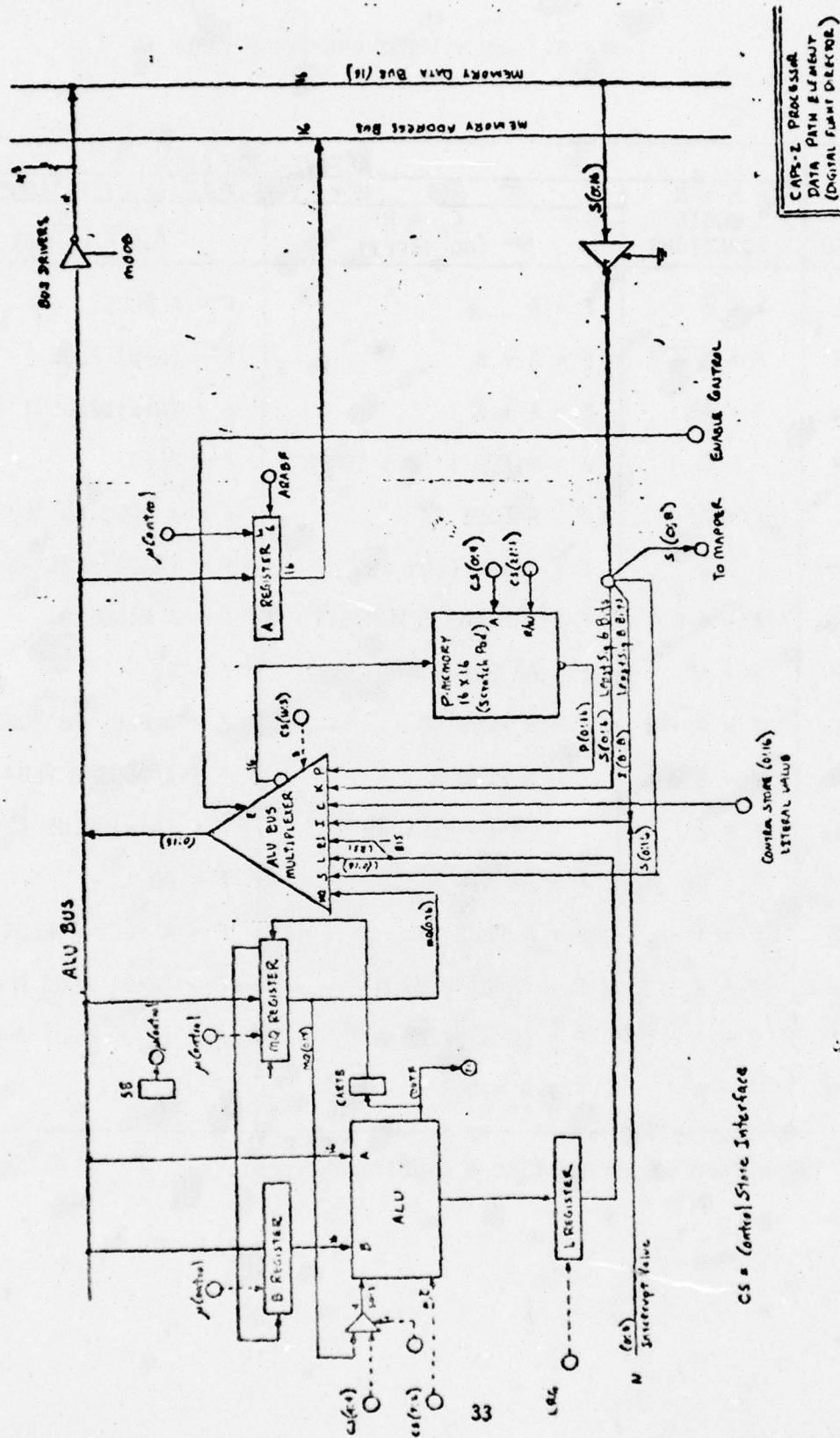


Figure 11. Data Path Logic

CAPS-2 PROCESSOR
DATA PATH ELEMENT
(DIGITAL FLIGHT DIRECTOR)

Rev. 6/78
J. K. S. 5/11/78

CS = Control Store Interface

TABLE 8. ALU ELEMENT OPERATION TABLE

SELECTION S3 S2 S1 S0	ACTIVE-HIGH DATA		
	M = H LOGIC FUNCTIONS	M = L; ARITHMETIC OPERATIONS	
		$C_n = H$ (no ⁿ carry)	$C_n = L$ (with ⁿ carry)
L L L L	$F = \bar{A}$	$F = A$	$F = A \text{ PLUS } 1$
L L L H	$F = \overline{A + B}$	$F = A + B$	$F = (A+B) \text{ PLUS } 1$
L L H L	$F = \bar{A}B$	$F = A + \bar{B}$	$F = (A+\bar{B}) \text{ PLUS } 1$
L L H H	$F = 0$	$F = \text{MINUS } 1 \text{ (2's COMPL)}$	$F = \text{ZERO}$
L H L L	$F = \bar{A}\bar{B}$	$F = A \text{ PLUS } \bar{A}\bar{B}$	$F = A \text{ PLUS } \bar{A}\bar{B} \text{ PLUS } 1$
L H L H	$F = \bar{B}$	$F = (A+B) \text{ PLUS } \bar{A}\bar{B}$	$F = (A+B) \text{ PLUS } \bar{A}\bar{B} \text{ PLUS } 1$
L H H L	$F = A \oplus B$	$F = A \text{ MINUS } B \text{ MINUS } 1$	$F = A \text{ MINUS } B$
L H H H	$F = \bar{A}\bar{B}$	$F = \bar{A}\bar{B} \text{ MINUS } 1$	$F = \bar{A}\bar{B}$
H L L L	$F = \bar{A} + B$	$F = A \text{ PLUS } AB$	$F = A \text{ PLUS } AB \text{ PLUS } 1$
H L L H	$F = \overline{A \oplus B}$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B \text{ PLUS } 1$
H L H L	$F = B$	$F = (A+\bar{B}) \text{ PLUS } AB$	$F = (A+\bar{B}) \text{ PLUS } AB \text{ PLUS } 1$
H L H H	$F = AB$	$F = AB \text{ MINUS } 1$	$F = AB$
H H L L	$F = 1$	$F = A \text{ PLUS } A^*$	$F = A \text{ PLUS } A \text{ PLUS } 1$
H H L H	$F = A + \bar{B}$	$F = (A+B) \text{ PLUS } A$	$F = (A+B) \text{ PLUS } A \text{ PLUS } 1$
H H H L	$F = A + B$	$F = (A+\bar{B}) \text{ PLUS } A$	$F = (A+\bar{B}) \text{ PLUS } A \text{ PLUS } 1$
H H H H	$F = A$	$F = A \text{ MINUS } 1$	$F = A$

*Each bit is shifted to the next more significant position.

Operations are affected by operating on data from the B register and ALU data bus. The outputs of the ALU can be stored in the L or A registers.

2.5.3.1.3 MQ, B, L, AND A REGISTERS

In addition to a scratch-pad memory, several registers are used for temporary storage of results. They are not visible to the macroprogram, only to the microprogram. Low power shottky register devices (54L194) are used for register functions. Register functions are described below:

MQ Register:

The MQ register is used as a utility register in the execution of instructions. Its primary value is in the execution of the multiply and divide instructions, hence the name MQ (for multiplier/quotient). MQ can be parallel loaded or shifted right or left one bit at a time. Its primary uses can be cataloged as follows:

1. Multiply

MQ holds the multiplier. Bit MQ(0:1) is examined to determine if the multiplicand in T is to be added to the partial product (PP). MQ is shifted right one bit along with PP. The least significant bit of PP (ALU output) is shifted into the msb of MQ. MQ holds the completed product after 16 shifts.

2. Divide

MQ hold the quotient bits as they are generated. MQ is shifted left one bit and the quotient bits are shifted into bit 0 each step.

3. Boolean

In the BOOL opcode, MQ(0:4) contain the "opcode" as defined elsewhere and as retrieved from the stack. The bits are used to directly control the ALU.

Closely associated with the MQ register is the SB (sign buffer) flip flop. During execution of the multiply instruction, it holds the original sign of the multiplier (the multiplier is "shifted-out" during execution). SB is used to determine if the multiplicand is to be added to or subtracted from the partial product. During division, SB holds first the sign of the dividend. SB is then EOR'ed with the divisor sign to hold the sign of the finished quotient. DIV is affected with all positive numbers. At the completion of the algorithm, SB determines if the quotient is to be complemented.

B-REGISTER:

This register is driven from the ALU data bus. The output provides one of the operands to the arithmetic and logic unit. The B-register can be parallel loaded or shifted right or left one bit at a time under microprogram control. The least significant eight bits of the B-register are also used for temporary storage of instruction literals during execution of the instruction fetch microroutine.

L-REGISTER:

The L-register provides temporary storage of arithmetic and logical results from the ALU. It is operated under microprogrammed control.

A-REGISTER:

The A (address) register is loaded from the ALU bus under microprogrammed control. The register is used to address main memory and I/O devices. The outputs are enabled to the address bus only after the CPU has been granted bus service.

2.5.3.1.4 SCRATCHPAD (P-MEMORY):

The P-memory is a 16 word bipolar memory. It is not accessible by software (user macroprograms). Addresses to P-memory come only and directly from the control store under microprogrammed control.

P-memory is used for utility storage during execution of micro-routines, for storage of pointers defining the current program, and for storage of constants and other special variables. Pointers in this memory are SPCR, TOS, and LENV. These pointers are initialized during the microprogrammed power-on-clear sequence.

Special stack registers in the P-memory are used as extensions of program stacks in main memory. Their use reduces the number of memory accesses that would be required if the stacks were entirely located in main memory.

2.5.3.2 MICROCONTROLLER

The advantages of microprogrammed control are well documented. It provides a disciplined, low cost approach to control logic design. In past years, it has been used extensively in large computers to provide a common architecture in machines covering a wide range of cost and capability. More recently, it has been used in minicomputers to customize an instruction set to a particular application.

Writing the microprogram is very similar to writing a problem program. However, the microprogram is like the hardware control logic that would be required in a non-microprogrammed computer in that it is implemented in read-only memory and is not altered after the machine has been built. Thus, the microprograms might be called "firmware" because they are designed like software and built like hardware. The DFDC microprogram is contained in Appendix I.

The microcontroller in the digital flight director provides a flexible, cost effective control logic design that allows addition of, modifications to, or deletion of instructions or operations using programming techniques.

2.5.3.2.1 BASIC OPERATION

The microcontroller consists of four parts: clock generation control store sequencer, control store and decode logic, and branch control logic. Figure 12 shows the basic structure of the microcontroller.

The function of the microcontroller is to provide all of the necessary control signals for the ALU data path section. It contains sufficient decision making logic to allow proper sequential operation in the data flow.

A microsequence is initiated by parallel loading the μ IAC (micro instruction address counter) with a control store starting location. The IAC is incremented through microsteps (or parallel loaded with branch address vectors if a branch condition exists) causing the emission of a new set of control signals for each microstep. A microsequence is normally terminated by branching to a machine control microsequence which causes the fetch of a new instruction of execution of an interrupt microsequence if an interrupt is pending.

2.5.3.2.2 CLOCK GENERATION

The basic microcycle time period is 250 nanoseconds. An eight megahertz clock is divided by two to achieve the resulting four megahertz microcycle frequency.

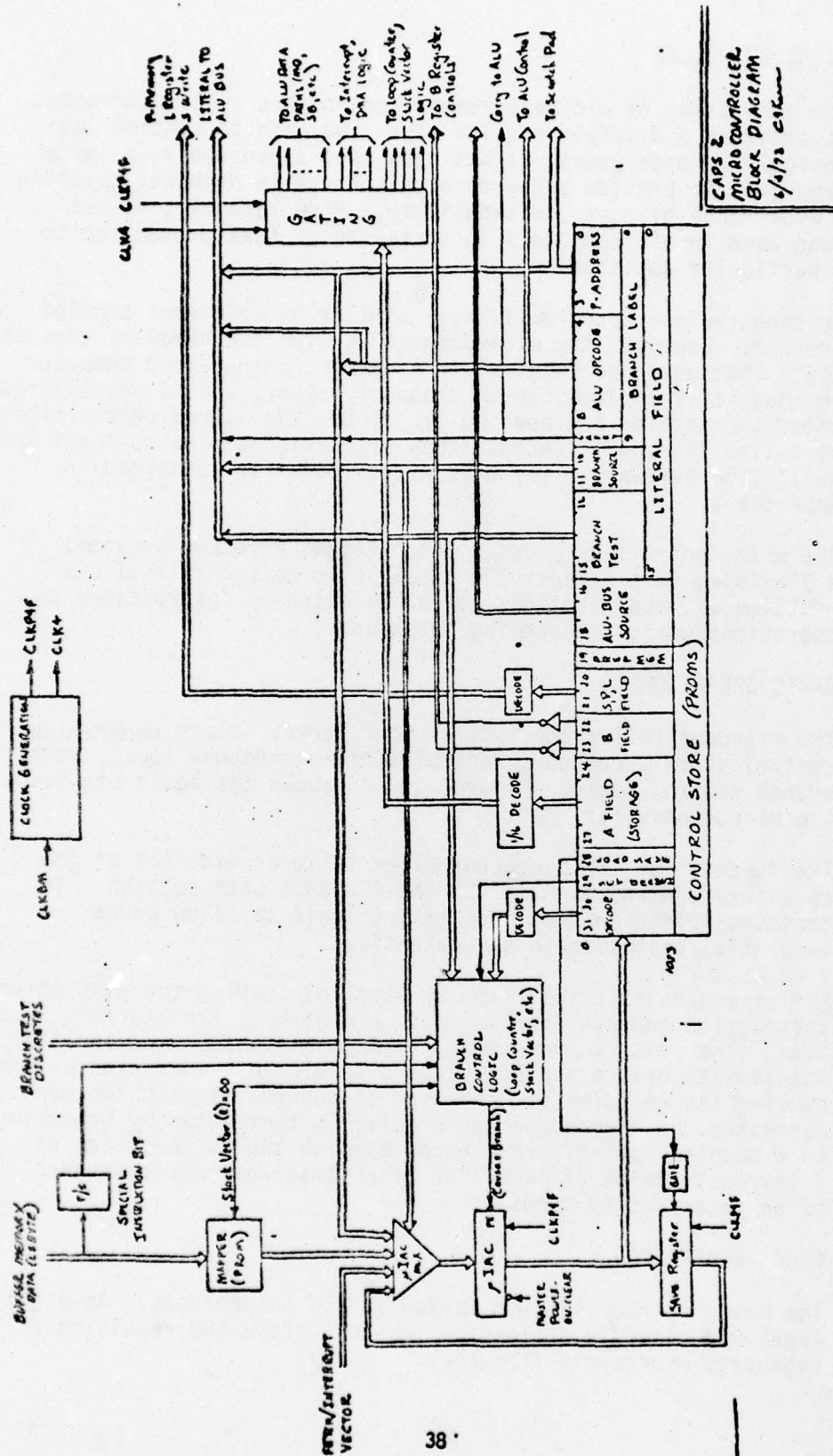


Figure 12. Microcontroller

All microcontrolled state changes or data storage occur during the last 25% of a microcycle. This portion of the microcycle is called clock phase four and the signal used to change state or load data has the mnemonic CLKP4F.

Figure 13 illustrates the basic clock signals and microcycle states. A microcycle is initiated by CLKP4F (μ IAC is parallel loaded or incremented with the leading edge of CLKP4F). After sufficient delay time through the μ IAC and control store memory, proper control signals are available. Referring to Figure 13, valid data appears on the ALU data bus after some delay time through the ALU data bus multiplexer.

Operations on and destinations for ALU data are determined by appropriate control store outputs. Destination storage occurs using CLKP4F. Most of the storage occurs synchronously with the leading edge of this clock signal. However, the scratch pad memory is written into using the entire duty cycle of CLKP4F.

2.5.3.2.3 CONTROL SEQUENCER

Referring to Figure 12, the control store, sequencer consists of an instruction mapper, an instruction address counter (μ IAC), a SAVE register and a μ IAC multiplexer.

The μ IAC outputs constitute the control store address vector. The counter can be incremented, cleared to zero (used to initiate a power-on-clear microsequence), or parallel loaded with a branch vector.

The mapper translates instruction syllables into control store address vectors. These branch vectors represent control store starting locations for microsequences required to execute individual instructions.

The SAVE register is used to provide subroutine capability at the microroutine level. It is loaded with the current control store address vector if desired and then is used as a branch return vector.

The μ IAC multiplexer is used to select the next control store address vector in the event a branch condition exists in the current state of the microcontroller. This vector can be selected from any of the following sources:

1. Mapper
2. SAVE Register
3. Machine Operation Vector (Fetch or Interrupt)
4. Control Store Output (Branch Vector)

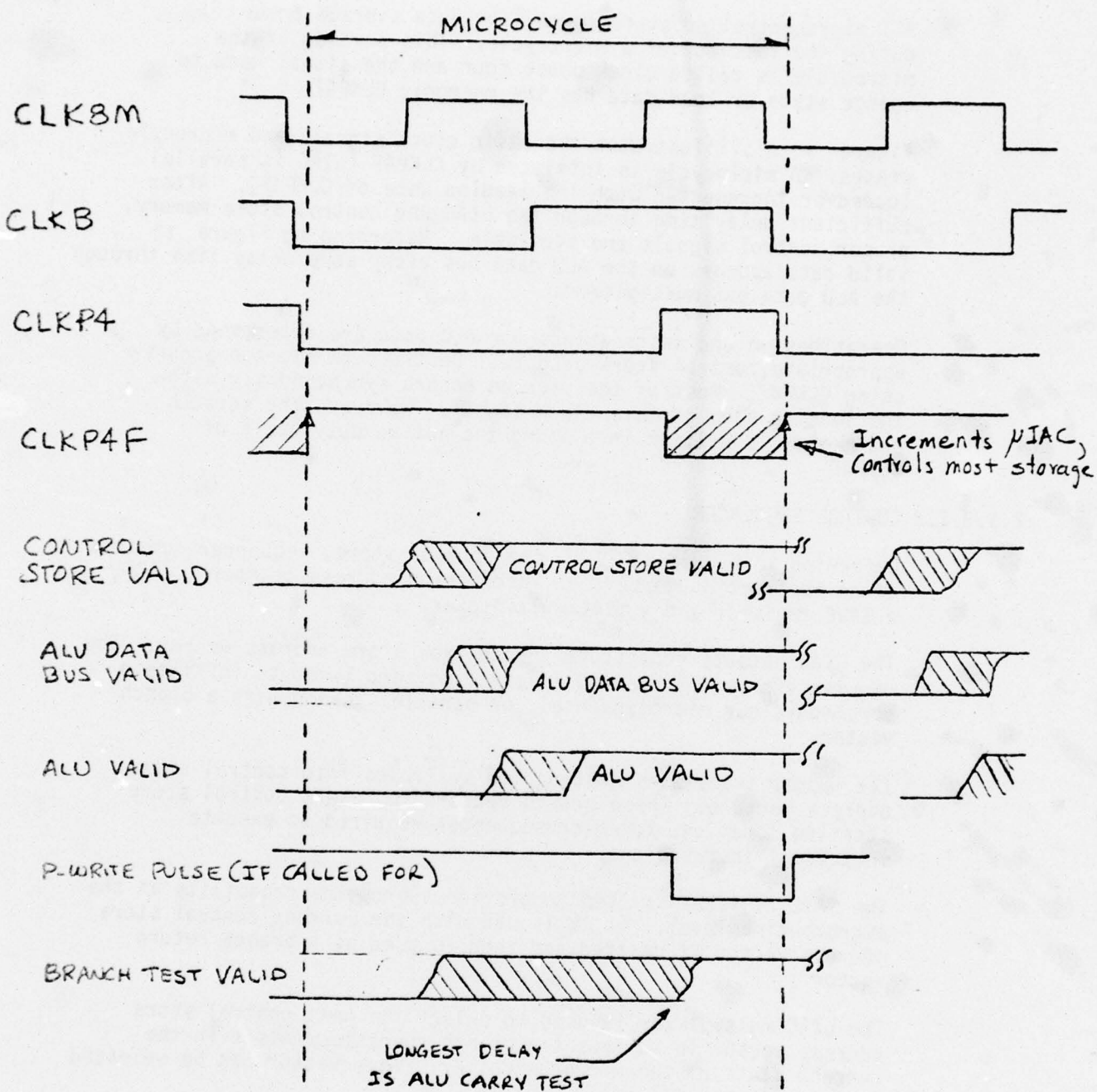


Figure 13. Basic Microcontroller Timing

2.5.3.2.4 CONTROL STORE AND DECODE LOGIC

Data paths in the computer are controlled by control store memory outputs. Certain groups of control signals are mutually exclusive and decode is used to provide those individual control signals. This results in a savings of memory. Figure 14 shows control store mapping. The assignment of bit positions has been arranged to ease reprogramming by assigning each field where possible, to specific PROM package columns. Design and space is provided for 1024 words of control store, each word being 32 bits in length.

Figure 15 details the microprogram field definitions. This figure and the mapping shown in Figure 14 provide sufficient reference information to prepare the microcode. The microcode (written in mnemonic form) is then supplied as data to a microassembler. The microassembler converts these mnemonics to control store bit patterns and assigns addresses to each step of the microprogram. The microassembler also converts branch labels to control store addresses, providing labeled microprogram entry points.

The following paragraphs (and Figure 14) briefly describe microprogram fields and mnemonic meanings:

Opcode Field (2 Bits)

This two-bit field is used to determine the next control state and to define alternate uses for dependent fields.

Definitions of operations specified by the opcode field are:

BNU (Branch Unconditionally): μ IAC is to be paralleled loaded with a branch address vector at the end of the current microcycle. The source of this vector is determined by the branch source field.

BCT (Branch Conditionally, True): Same operation as BNU except that the branch will occur only if the branch test flip flop is true (set).

BCF (Branch Conditionally, False): Same as BCT except criteria is a false condition.

INC (Increment): μ IAC is to be incremented.

Field 1: OPCODE (Bits 30-31)

BCF...00 Conditional Branch if False
 BCT...01 Conditional Branch if True
 BNU...10 Unconditional Branch
 INC...11 Increment uIAC

Field 2: BRANCH SET (Bit 29)

SBR...0 Set Branch (Test)
 NTS...1 NOP Condition

Field 3: SAVE Register Field (Bit 28)

SAV...0 Load SAVE Register with uIAC
 NSV...1 NOP Condition

Field 4: A FIELD (Bits 24-27)

STA...0000 Load Memory Address Register
 MCP...0001 Load MQ Register
 MQM...0010 Load Most Significant Byte of MQ
 MQR...0011 Shift MQ Right One Bit
 MLQ...0100 Shift MQ Left One Bit
 CLD...0101 Load Loop Counter from ALU Bus
 CTL...0110 Increment Loop Counter
 IST...0111 Set Microprogrammed Interrupts
 IRS...1000 Reset Interrupts (Identity in L)
 UMT...1001 Toggle User Mode Flip Flop
 SBL...1010 Load Sign Bit Flip Flop
 SPO...1011 EOR Load to Sign Bit Flip Flop
 POP...1100 POP Stack Vector
 PSN...1101 PUSH Stack Vector
 SPR...1110 TOGGLE Spare Flip/Flop
 NOT...1111 NOP Condition

Field 5: B FIELD (Bits 22-23)

BRC...00 Load B Register
 BRG...01 Shift B Right One Bit
 BLS...10 Shift B Left One Bit
 NOP...11 NOP Condition

Field 6: S,P,L Field (Bits 20-21)

PVR...00 Load P Memory
 SVR...01 Load S Memory
 LRG...10 Load L Register
 NOM...11 NOP Condition

Field 7: PREPARE MEMORY ACCESS (Bit 19)

PRP...0 Next ucycle is Memory Access
 MAN...1 NOP Condition

Field 10: ALU Bus Source (Bits 16-18)

LSHR...000 Shift L Right to ALU Bus
 LRRR...001 Put L Register on ALU Bus
 TRGR...010 Put H and MSByte on ALU Bus
 MQRG...011 Put M Register on ALU Bus
 MEMR...100 Put Least 6 bits of Memory on ALU Bus
 SMEM...101 Put S Memory on ALU Bus
 CNST...110 Put Literal Value on ALU Bus
 PMEM...111 Put P-memory on ALU Bus

Field 11: BRANCH CONTROL (Bits 12-15)

LCTR...0000 Test Loop Counter for 1111(False)
 MQGO...0001 Test MQ Register Bit 0
 USFR...0010 Test User Mode Flip Flop
 BRNG...0011 Test B Register for Negative Value
 SBIT...0100 Test Special Instruction Bit
 SBIT...0101 Test Sign Bit Flip Flop
 COUP...0110 Test Carry Out from ALU (True Test)
 ANEG...0111 Test ALU for Negative Condition
 LNEG...1000 Test L Register for Negative Value
 STVO...1001 Test Stack Vector = 00
 STVL...1010 Test Stack Vector = 1x
 AEQO...1011 Test A = B (From ALU)
 ZERO...1100 Reset Test Flip Flop
 IPPY...1101 Test for Interrupt Condition
 MQOS...1110 Test MQ, Bit 5
 SETL...1111 Set Test = Logical "one"

NOTE: All fields not called out by the microprogram will contain "1"s.

NOTE: Every mnemonic or label must be unique--no duplications are allowed.

Field 12: BRANCH SOURCE (Bits 10-11)

MAP...00 Branch Vector = Mapper
 FCH...01 Branch Vector = Fetch/Interrupt
 BSV...10 Branch Vector = SAVE Register Value
 BOM...11 Branch Vector = Control Store, Branch Label

NOTE: Field 12 is also used for ALU transfer and for rotate functions. These are defined in Fields 13 and 14. Do not branch when using them.

Field 13: ALU TRANSFER (Bit 11)

T...0 Transfer Control of ALU to MQ Register
 M...1 Microcontrol of ALU

Field 14: ROTATE (Bit 10)

R...0 Rotate Data (Used with shift operations)
 S...1 No Rotate

Field 15: CARRY (Bit 9)

C...0 Carry to ALU
 N...1 No carry

Field 16: ALU OPCODE (Bits 4-8)

PASS...00000 A (If carry, add one to A)
 OR...00001 A OR B
 ORBN...00010 A OR NOT B
 NEG1...00011 Minus 1
 A4...00100 A Plus A5
 A5...00101 (A+B) Plus A5
 SUBJ...00110 A Minus B Minus 1
 A7...00111 A5 Minus 1
 AB...01000 A Plus AB
 ADD...01001 A Plus B
 A10...01010 A+5 Plus AB
 A11...01011 AB Minus 1
 AA...01100 2A
 A13...01101 A+B Plus A
 A14...01110 A+B Plus A
 DEC...01111 A Minus 1

NOTA...10000 X
 NOR...10001 A-B
 L1B...10010 AB
 L2R...10011 0
 MAND...10100 AB
 NOTB...10101 B
 EOR...10110 A-B
 L23...10111 AB
 L24...11000 A-B
 NEOR...11001 A-B
 B...11010 B
 AND...11011 AB
 LOG1...11100 Logical One's
 L29...11101 A+5
 LOR...11110 A+B
 A...11111 A (Not affected by carry bit)

Field 17: P ADDRESS (Bits 0-3)

TO...0000 Utility Register
 ONE...0001 Integer 1
 T1...0010 Utility Register
 T2...0011 Utility Register
 T3...0100
 T4...0101
 T5...0110
 MSK1...0111 Interrupt Mask (Integer Values 7)
 T6...1000 Utility Register
 SPCR...1001 Syllable Pointer
 TOS...1010 Stack Pointer
 LENV...1011 Local Environment Pointer
 PD...1100 Utility Register
 MSKL...1101 Mask for Reference Index values
 SRA...1110 Stack Register a
 SRB...1111 Stack Register b

Field 18: BRANCH LABEL (Bits 0-9)

Bits 0-9 are used to supply a branch vector if the branch source is BOM. Otherwise they either make up the ALU opcode and P-address or form part of the literal field.

Field 19: LITERAL (Bits 0-15)

This field is the source of the literal put on the ALU bus if CNST is called as the bus source. All microprogram entries starting with a numerical value will be treated as literals.

Figure 15. Microprogram Field Definitions

Branch Set Field (1 Bit)

SBR (Set Branch): The branch test flip flop is to be set if the condition specified by the branch select field is true.

NTS (No Test): NOP Condition.

SAVE Register Field (1 Bit)

SAV: Load the SAVE register with the current value of μ IAC.

NSV (No Save): NOP Condition

A Field (4 Bits)

This field allows one of the following storage or state change operations:

- Load address register
- Load M
- Load address register
- Load MQ register
- Load most significant byte of MQ register
- Shift MQ register left one bit
- Shift MQ register right one bit
- Load loop counter
- Increment loop counter
- Set microprogrammed interrupts
- Rest selected interrupt
- Toggle user mode flip flop
- Load sign bit flip flop
- Load sign bit flip flop (EOR)
- Reset stack vector bit 0
- Reset stack vector bit 1
- Set stack vector
- NOP (No Operation)

B Field (2 Bits)

This field controls the operation of the B register. The B register can be loaded from the ALU data bus, shifted left one bit, or shifted right one bit.

S, P, L Field (2 Bits)

This field is used to cause loading of the L register, a P-memory write, or an S-memory write.

Memory Access Field (1 Bit)

If this bit is a zero, a memory access is to be initiated immediately after completion of the current microcycle.

ALU Bus Source Field (3 Bits)

This field provides direct encoding to the memory bus multiplexer to gate data to the ALU data bus. The following are possible sources for ALU bus data:

- MQ Register
- L Register (ALU outputs)
- L Register shifted left one bit
- L Register shifted right one bit
- P-Memory
- Least significant 6 bits of memory
- Literal value from control store
- Interrupt value (N) and least significant byte to most significant byte on ALU bus.

Branch Control Field (4 Bits)

This field selects the branch test condition. If a set branch condition exists, the branch condition flip flop will assume the boolean value of the test selected.

Branch Source Field (2 Bits)

This field selects the source of the branch vector. It also serves as ALU control transfer and rotate control (valid only if opcode is INC). The following branch vectors may be selected:

- Control Store (Branch Label)
- Mapper (Instruction starting address)
- SAVE Register
- Instruction Fetch/Interrupt Vector

Carry and ALU Opcode Fields (6 Bits)

Six bits are used to control arithmetic logic functions as described in section 2.5.3.1.2.

P-Address (4 Bits)

Four bits are used to supply the P-memory (scratch pad, pointers) address. This field, together with the ALU opcode field, also form the branch address in a branch step.

Branch Label (10 Bits)

If a branch opcode exists and the branch vector source is ROM (control store), then this field provides the branch vector. Otherwise, this field is the ALU opcode and P-address fields.

2.5.3.2.5 BRANCH CONTROL LOGIC

The functions of the branch control logic are as follows:

1. Select the branch test
2. Provide a counter for microprogram looping
3. Stack vector control
4. Control the parallel loading of the μ IAC on branch operations.

Figure 16 shows the functional concept of this logic. Branch test conditions are fed into tri-state multiplexers and are selected under control of the branch control field of the control store. If an SBR (set branch) condition exists, the branch condition flip flop will be set with the results of the selected test.

PEF signal controls the parallel loading of the μ IAC. This gate is forced to a logical zero if there is an unconditional branch or is a zero conditionally on the state of the branch condition flip flop.

The loop counter is loaded under microprogrammed control from the ALU data bus. The TC output is a 1 when an all one's condition exists in this counter. This condition is testable.

Flip flops E0 and E1 are used as stack register indicators to show if stack data is stored in the scratch pad.

2.5.3.2.6 EXAMPLE MICROSEQUENCE

As an example of a microprogram, consider the instruction fetch operation. The microprogram might look like this:

<u>Step</u>	<u>Label</u>	<u>Microprogram</u>
1	INTERP	PMEM SPCR STA PASS C LRG
2		LRGR PWR SPCR PRP
3		BNJ MAP KMEM MQP BRG

Explanation of the steps are as follows:

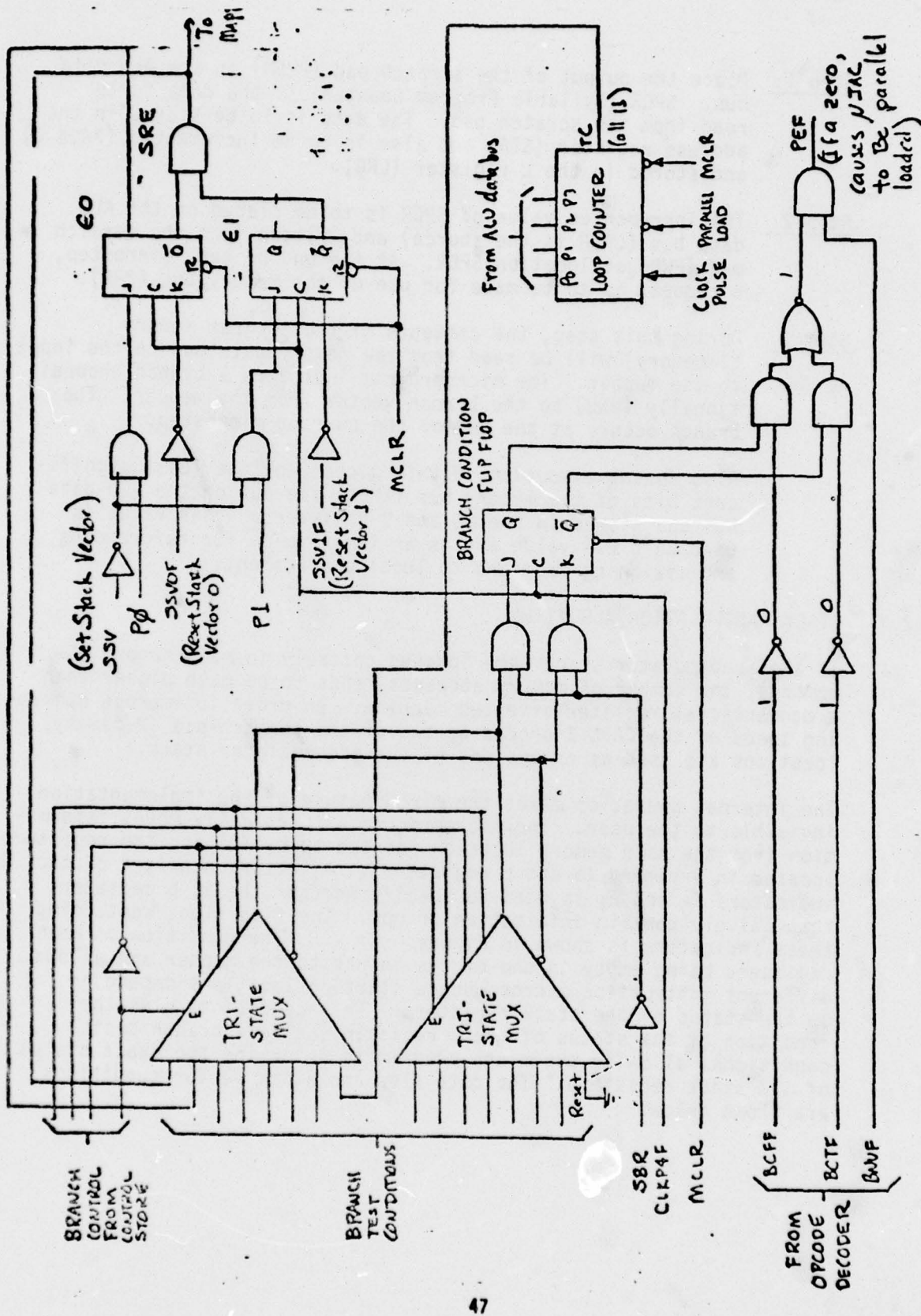


Figure 16. Branch Control Logic

6/6/73

Step 1: Place the output of the scratch pad (PMEM) on the ALU data bus. SPCR (Syllable Program Counter) is the data to be read from the scratch pad. The data is to be stored in the address register (STA) and also is to be incremented (PASS C) and stored in the L register (LRG).

Step 2: The incremented value of SPCR is to be placed on the ALU data bus (LRGR is the source) and written into the scratch pad (PWR) at location SPCR. At the end of this microstep, a request is to be made for use of the memory bus (PRP).

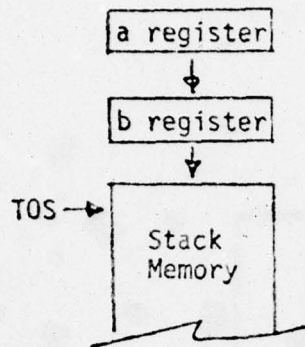
Step 3: During this step, the contents of the program memory (I-memory) will be read from the memory data bus to the input to the mapper. The microprogram indicates a branch unconditionally (BNU) to the branch vector from the mapper. The branch occurs at the end of the current microstep.

Also during execution of this microstep, the least significant bits of the memory bus (KMEM) are put on the ALU data bus and stored in the MQ and B registers. This value is used as a LIT value and as an index value for referencing and assigning from and to local environment.

2.5.3.2.7 STACK MANIPULATION ALGORITHM

If the accumulator stack were located entirely in main memory (S-memory), the number of memory accesses tends to be much higher than a conventional register oriented machine. In order to improve operating speed of the CAPS 2 processor two of the scratch-pad (P-memory) locations are used as extensions of the accumulator stack.

The internal operation makes the mixed nature of the implementation invisible to the user. The microprogram automatically moves information from the main memory location pointed to by TOS and the registers located in P-memory (a and b registers). A vector comprised of two indicators E_0 and E_1 is used to specify whether the a, b registers respectively contain information or not. The flip flops containing these indicators is shown in Figure 16. The condition of both registers being empty is one of the inputs to the mapper and allows different instruction microsequence starting locations depending on the status of the stack registers. This condition, plus the condition of the status of the a register (E_0) are branch test conditions, allowing the micro program to determine the exact status of the stack registers. The data flow and stack vector conditions are shown below:



$E_0, E_1 = 00$ both registers empty

$E_0, E_1 = 01$ a empty, b full

$E_0, E_1 = 11$ both full

$E_0, E_1 = 10$ not allowed

2.5.3.3 TRANSFER BUS

The CAPS system is structured around a common bus, interfacing the processor, memory and I/O devices. This bus, called the Transfer Bus, provides the pathway for all data communications among all attached units. Both device-to-memory and device-to-device communications can take place on the bus.

2.5.3.3.1 BUS STRUCTURE

The Transfer Bus may be partitioned into two unrelated but dependent components: the data transfer busses and associated controls and the bus priority/contention logic. Figure 17 shows these two components. Figure 18 lists and defines the lines in each.

All data words (≤ 16 bits) are transferred in parallel over the DB lines. The address of the selected device or memory address is placed on the AB address bus. The controls IORQ, IORP and IOWR sequence the direction and timing of the transfer.

The BRQ (bus request) and SEIZE lines and the daisy-chained GRANT line provide control over which device is to have access to the transfer bus.

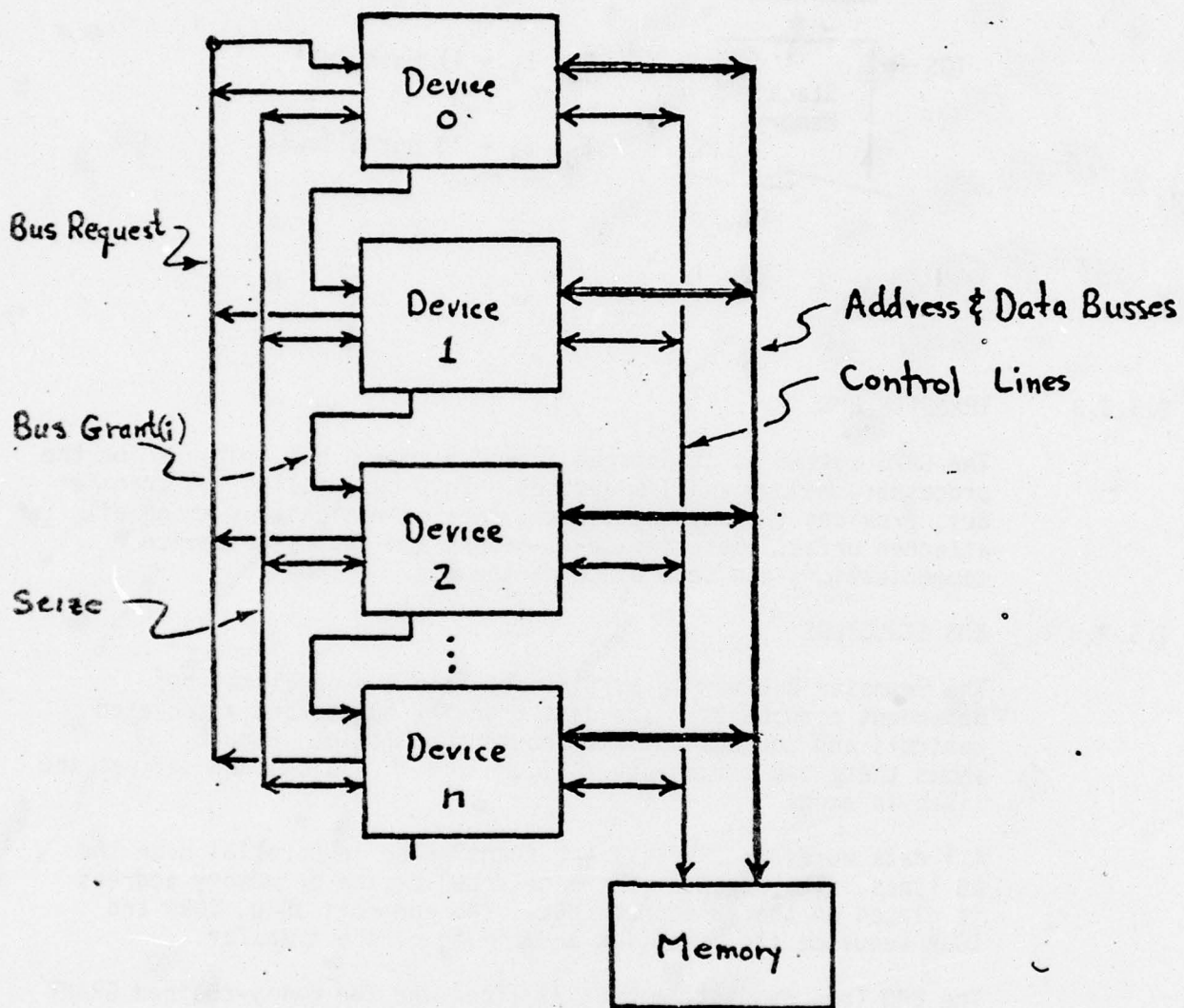


Figure 17. Transfer Bus Organization

<u>LINE</u>	<u>NO. OF LINES</u>	<u>ACTIVE</u>	<u>DEFINITION</u>
<u>DATA TRANSFER BUSS</u>			
\overline{DB}_i	16	LOW	Bi-directional data bus (I/O from "S")
\overline{AB}_i	12	LOW	Address bus
\overline{IORQ}	1	LOW	"Request" from originating device
\overline{IORP}	1	LOW	"Response" from addressed device
\overline{IOWR}	1	LOW	"Write Select"; indicates originating-to-selected device transfer
$\overline{CLK\&M}$	1	-	8 MHz clock for timing
\overline{DEVEN}	1	LOW	Device enable for direct access.
\overline{FAULT}	1	LOW	Invalid Address Interrupt
<u>PRIORITY/CONTENTION LOGIC</u>			
\overline{BRQ}	1	LOW	Bussed Device "Requests" (Bids) for use of DMA channel.
\overline{BG}_i	1 (per device)	HIGH	DME Channel "Grant" from higher-priority device to next lower-priority device.
\overline{SEIZE}	1	LOW	Device has received "Grant" and has "SEIZE'd" control of DMA channel; a "BUSY" flag.

Figure 18. Transfer Bus Line Definitions

2.5.3.3.2 DATA TRANSFER

Once a device has gained access to the bus interface, it has and can retain control of the interface until its needs are satisfied. It can transfer one or a dozen words before releasing control. All other devices are "locked out" during this time.

The controlling (or originating) device places the address of the requested device or memory location on the address bus (AB). It selects the direction of transfer of data by setting IOWR appropriately. IOWR = 1 selects data out of the originating device. IORQ is set when all is ready. IORP is monitored for a response from the selected device. Then IORQ is reset and the data and address lines cleared. Figure 19 gives the timing.

Critical points are labelled. Point A shows the address and IOWR lines being set up before IORQ is set at B. The final transition of IORP at C indicates write data is accepted or read data is available on the DB. Point D is acknowledgment of the IORP. At this time address and data are removed.

There are no timing restraints on exactly when transitions should occur. Only the relative timing as shown in the figure is critical. Also IORQ and IORP are switched in synchronism with (8MHz) clock transitions.

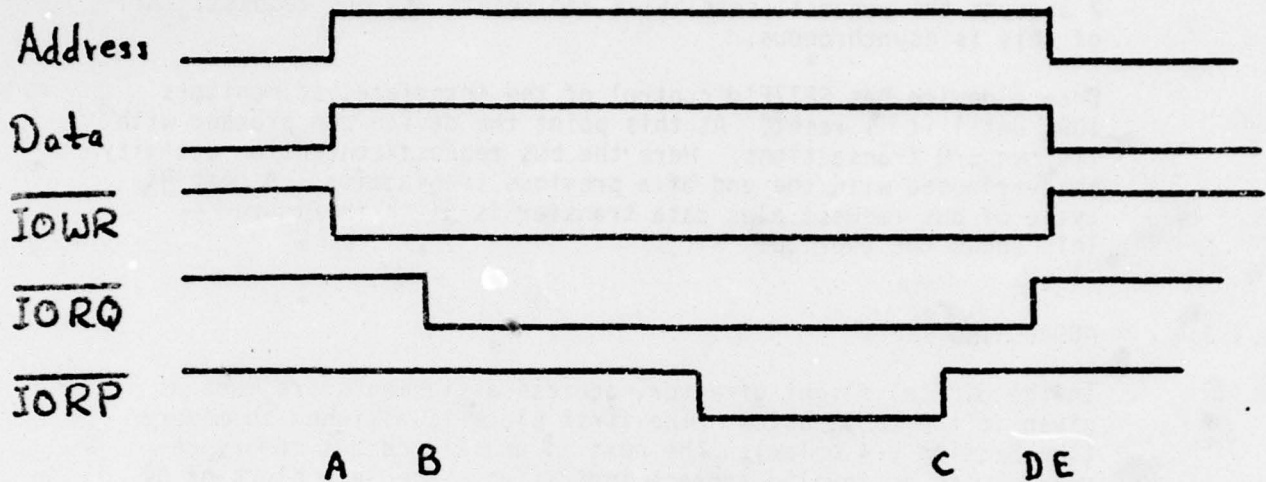
2.5.3.3.3 PRIORITY/CONTENTION LOGIC

In order for a device to transfer data on the bus, it must first gain access to the bus. This is affected by the bus priority and contention hardware.

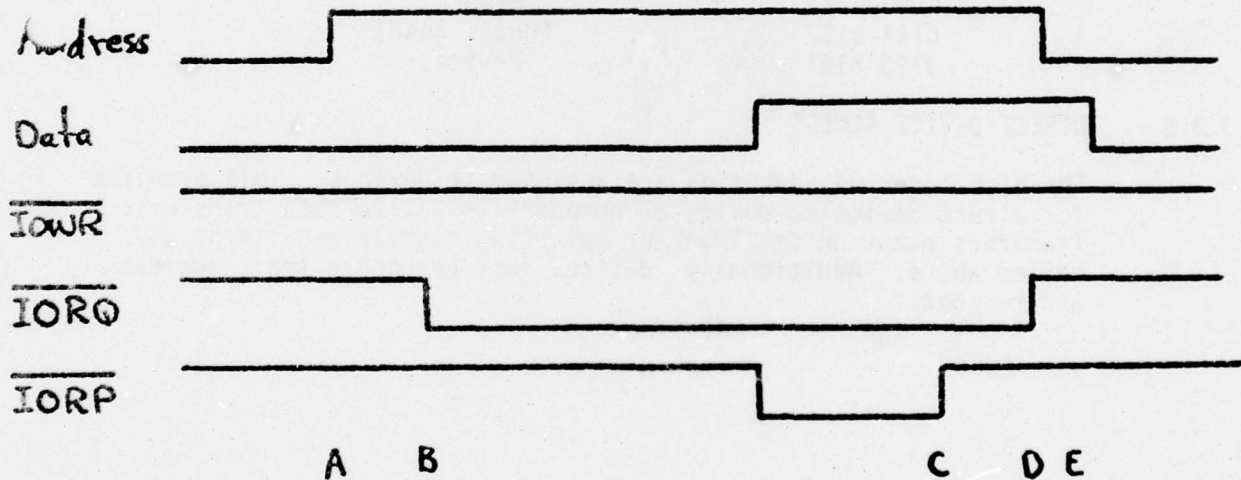
When a device requires use of the direct memory access interface, it first sets the Bus Request signal (Figure 17). This request is turned around and serially rippled down all the devices in the Bus Grant chain. The highest priority device receives the grant first. If it has no request pending, it forwards the grant to the next device in the chain. The 'grant' stops at a requesting device which then sets the 'SEIZE' bus active. This 'SEIZE' signals that a device now has control of the Transfer Bus interface. All further device requests are inhibited until the SEIZE signal is reset.

By resetting SEIZE before its last access on the Transfer Bus is complete the active device allows another request-grant cycle to complete before the access is complete. This overlap, then, causes most or all of the contention time to be masked out.

WRITE:



READ:



- A. SET-UP : Output Address, IOWR, Data
- B. INITIATE : Address Valid
- C. RESPONSE: Data Accept/Valid
- D. TERMINATE: Acknowledge / Data Accept
- E. Disconnect: Remove address, data

Figure 19. Transfer Bus Timing

Figure 20 shows the request-grant timing. Device 2 in the figure has set Bus Request at some point after SEIZE has been reset. Bus Grant ripples down to device 2 through devices 0 and 1. Device 2 accepts the request, sets SEIZE and resets its bus request. All of this is asynchronous.

Once a device has SEIZE'd control of the interface, it monitors IORQ until it is reset. At this point the device can proceed with its own I/O transactions. Here the bus request/contention activity is overlapped with the end of a previous transaction. A complete cycle of bus request plus data transfer is given in Figure 21. This shows the overlap.

2.5.3.3.4 ADDRESSING

In the digital flight director, address assignments are made as given in the table below. The first block is assigned to memory (see Section 2.4 below). The next is unassigned but memory responds with an invalid address indication. The last block of 64 words is assigned as (I/O) device register addresses.

<u>Address</u>	<u>Assigned To</u>
0000-6143	Memory
6144-8127	(Unassigned)
8128-8191	Devices

2.5.3.3.5 DIRECT DEVICE ACCESS

The high-order 64 addresses are assigned to devices. This provides for direct device-to-device or computer-to-device data transfers. Transfers occur on the Transfer Bus using signals and timing detailed above. Additionally, devices must recognize their address and respond.

TRANSFER BUS PRIORITY / CONTENTION TIMING

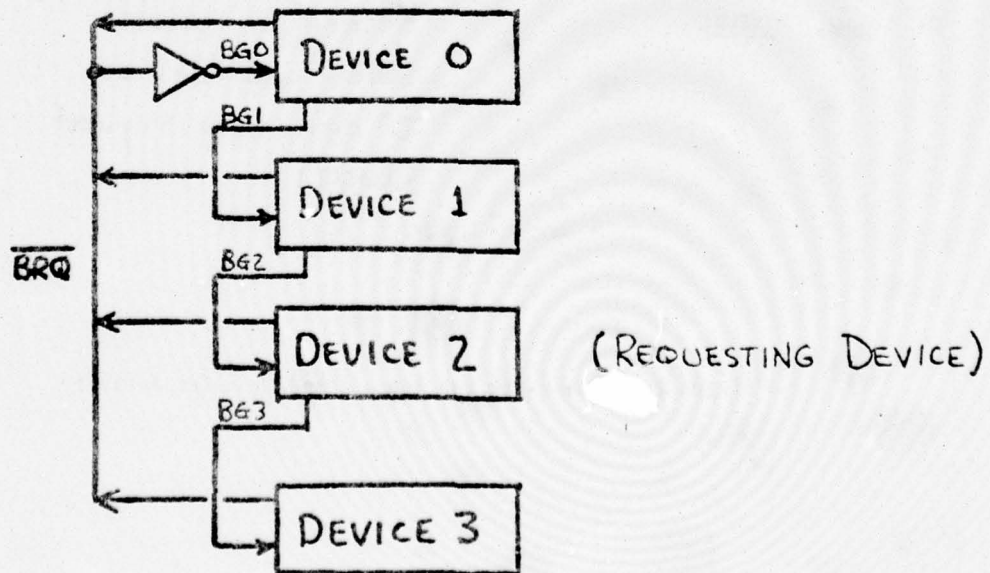
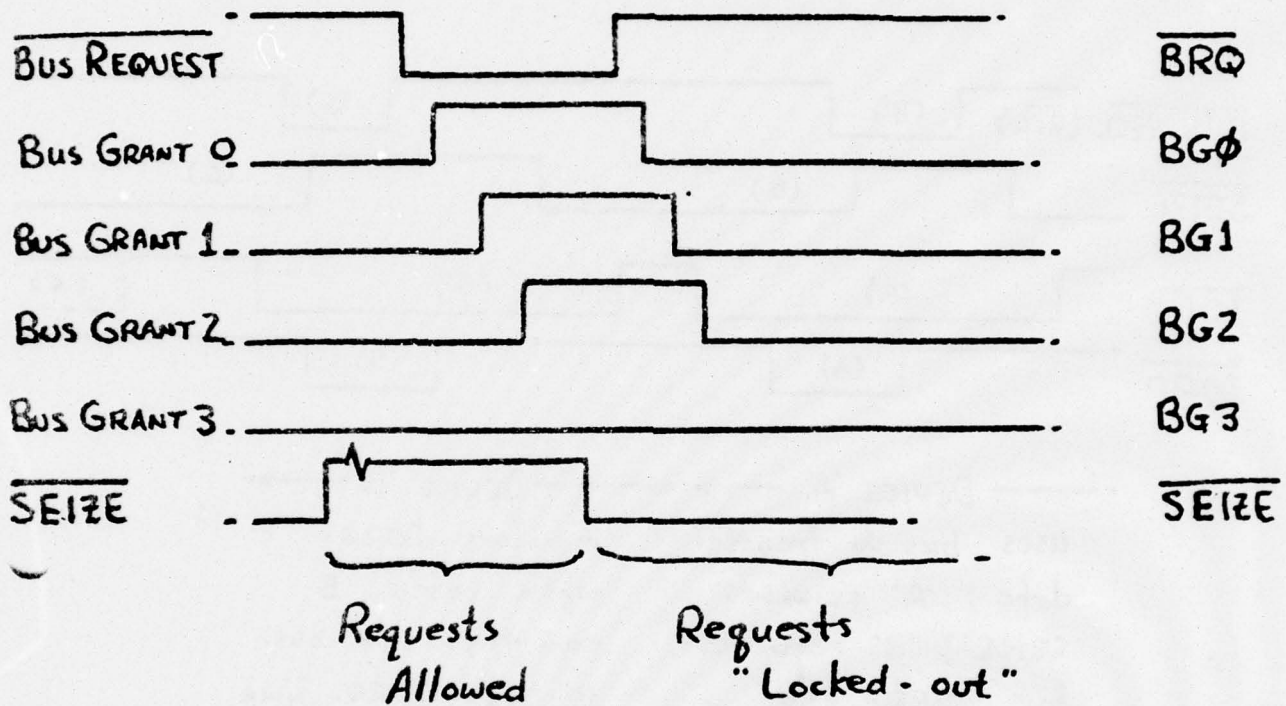
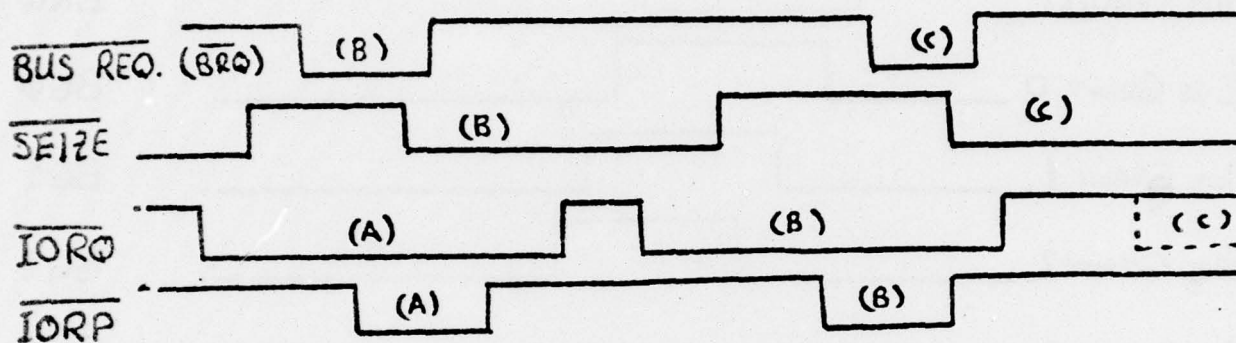


Figure 20. Transfer Bus Priority Timing



———— DEVICE 'A' ———— ← ———— DEVICE 'B' ————
 uses bus to transfer data ; 'A' releases SEIZE ; 'B' requests bus (BRQ) and receives GRANT and sets SEIZE
 monitors IORQ — when reset, B can then use bus. 'B' sets IORQ. Since a 2nd access is not needed B resets SEIZE and device 'C' can now request (BRQ)

Figure 21. Composite Bus Request and Transfer Activity

2.5.3.4 MEMORY

The memory function contains the working store, termed the S-Store (Stack), and the instruction, or I-Store. The S-Store is a 1024 word by 16 bit read/write RAM, and the I-Store is a 4096 word read-only store. The memory responds to read and write requests from all devices attached to the Transfer Bus System.

2.5.3.4.1 MEMORY STRUCTURE

The CAPS memory consists of the storage module, containing a mixture of RAM and ROM; I/O gating for the data; address decoding logic; control and timing circuitry; and RAM-refresh timing logic. Figure 22 shows how these functions are related. The storage module is covered in detail below; its structure and circuitry are discussed. The control provides read/write/refresh timing to the storage module and to the Transfer Bus interface. Refresh interval timing is required for the dynamic RAM's used in the storage module.

2.5.3.4.2 MAP

Figure 23 is a map of the storage module structure, showing data word widths and address assignments. The first 1024 words are 16 bits wide read/write random access store (RAM). The next block of 512 words is 16 bit wide read-only store (ROM) for storage of program constants. The third block of 4096 words is 8 bits wide, used for program (instruction) store.

A group of 1984 addresses (6184-8187) are unassigned. An access request to one of these results in an "invalid address" indication - a 'fault' interrupt to the processor. The remaining 64 words are reserved for assignment to I/O devices attached to the Transfer Bus.

2.5.3.4.3 MEMORY ELEMENT HARDWARE

Memory storage elements have been chosen with reliability, flexibility and power consumption as primary considerations. The 1024 word RAM is constructed from National Semiconductor MM4260 1024X1 dynamic MOS RAM's. These devices require under 100 mw per package during standby. Their power peaks at 500 mw only during pre-charge. Duty cycle in the flight director is 25% or less.

Program ROM is constructed from 512X4 programmable ROM's (PROM's). Each PROM requires about 600 to 800 mw in operation. Power switching is employed to significantly reduce power.

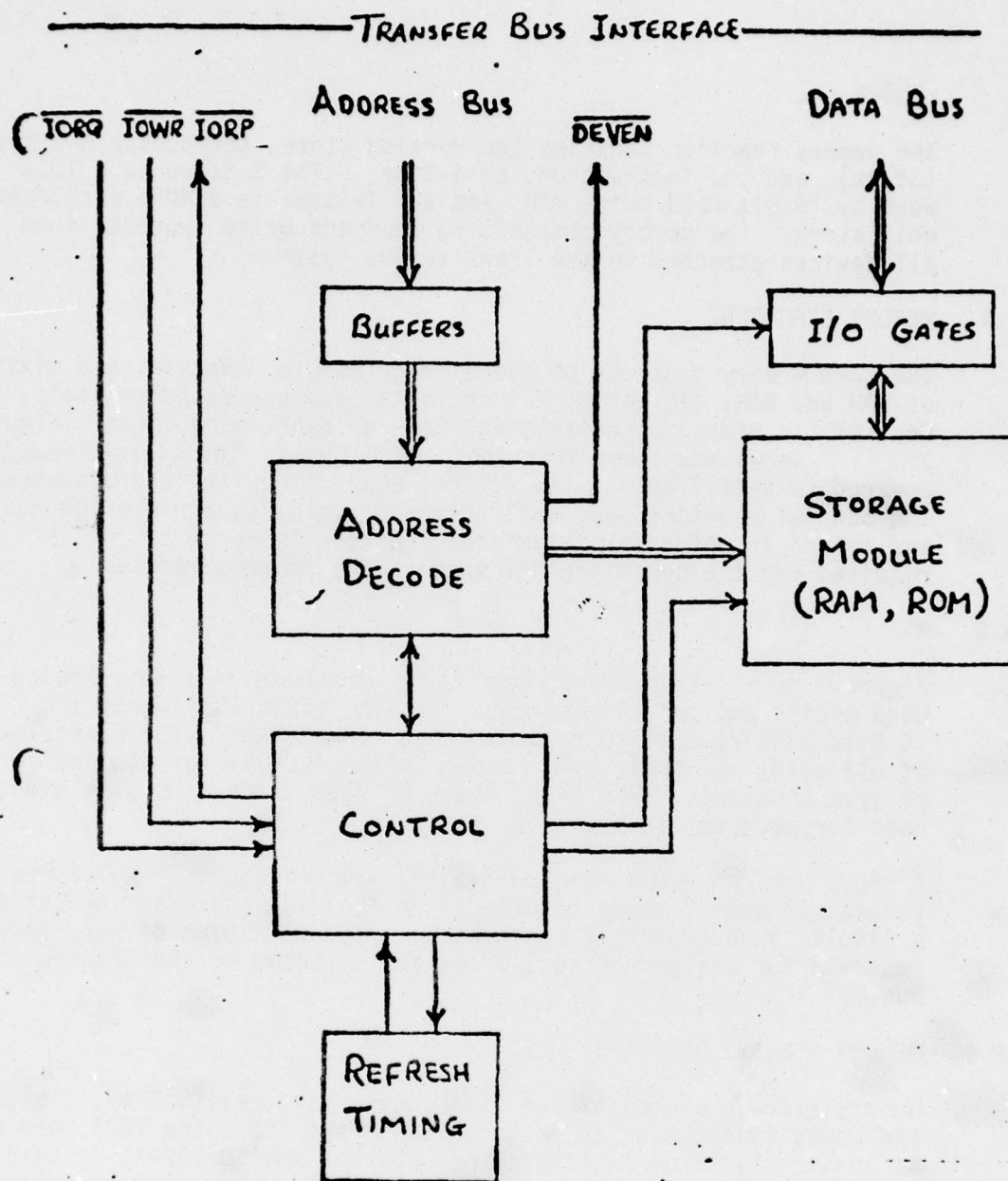


Figure 22. CAPS Memory Module

1024 WORD
S-STORE RAM

0000

0000

16 bits wide

1023

1777₈

2048

4000₈

16 bits wide

2557

4777₈

2560

5000₈

8 bits wide

4096 WORD
I-STORE ROM

8 bits wide

8 bits wide

6143

13777₈

UNASSIGNED

DEVICE ADDRESSES

8127

4031

8191

17677₈

17777₈

Figure 23. Storage Module Map

2.5.3.4.4 POWER SWITCHING

Without power switching, the 12 PROM's required would dissipate (12 X 800) 9.6 max. Figure 24 illustrates the memory as configured for power switching. Three more devices are required to switch the power, but power dissipation is significantly reduced. The switches turn the +5 volt power to the PROM's 'ON' only when required by an access request. When OFF, obviously, the PROM's consume no power. The switches, when OFF, require 30 mw per PROM. When ON, a switch (one per PROM) consumes 100 mw.

With power switching, when no I-Store access is being made, I-Store power is $.12 \times 30 = 360$ mw (vs 9600 mw).

Worst-case active power is:

10 X 30 mw = 300 mw (OFF switches)
4 X 100 mw = 400 mw (ON switches)
4 X 800 mw = 3200 mw (ON PROM's)

3900 mw maximum

2.5.3.4.5 MEMORY INTERFACE

The memory module interfaces with the Transfer Bus lines defined earlier. The memory is a passive device in the sense it can not originate any Transfer Bus access requests. It can only respond. Bus lines connected to the memory are listed below:

(S = Source D = Destination)

\overline{AB}_i	Address Bus	D
\overline{DB}_i	Data Bus	S/D
\overline{TORQ}	Request	D
\overline{TORP}	Reply	S
\overline{IOWR}	Write Mode	D
\overline{DEVEN}	Device Enable	S
\overline{POC}	Reset	D
$\overline{CLK8A}$	8 MHz Clock	D
\overline{FAULT}	Invalid Address	S

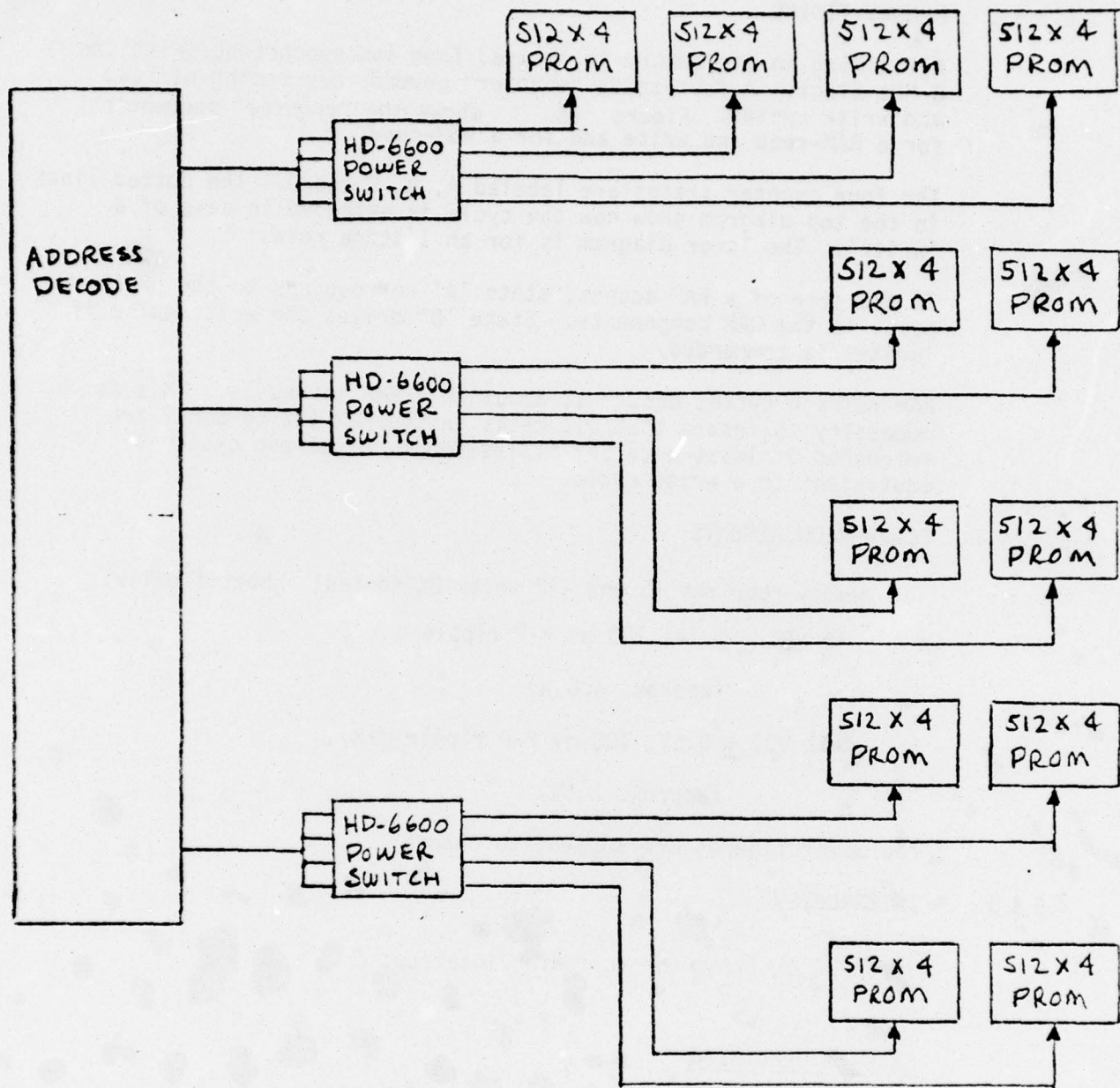


Figure 24. PROM Power Switching

2.5.3.4.6 MEMORY TIMING

All timing in the memory is derived from and synchronous with the 8 MHz clock. A four-state "counter" governs sequencing of read and write cycles. Figure 25 shows the "counter" sequencing for a RAM-read and write and for a ROM-read.

The four counter states are labeled A, B, C and D. The dotted lines in the top diagram show how the cycle is extended in case of a 'write'. The lower diagram is for an I-Store read.

In the case of a RAM access, state 'A' corresponds to the precharge pulse to the RAM components. State 'D' drives the write pulse if 'write' is commanded.

RAM refresh cycles occur at about 30 μ -sec intervals. This is necessary to insure that all cells in the RAM (32 columns) are refreshed at least once per millisecond. A refresh cycle is equivalent to a write cycle.

2.5.3.4.7 POWER REQUIREMENTS

The memory requires +5 and -12 volts DC nominal. Specifically:

+5 VDC \pm 0.2V, 100 mv P-P ripple max.,

(approx. 4.0 W)

-12 VDC \pm 0.5V, 100 mv P-P ripple max.,

(approx. 2.5W)

The power figures are subject to change.

2.5.3.5 INTERRUPTS

The DFDC utilizes the following interrupts:

Machine Generated

- Halt (Program initiated)
- Illegal Address
- Illegal Opcode

I/O Generated

- Time phase a
- Time phase b
- Data ready

The number of interrupts can be expanded externally. Interrupts are scanned, masked, serviced and reset under both microprogram and software control.

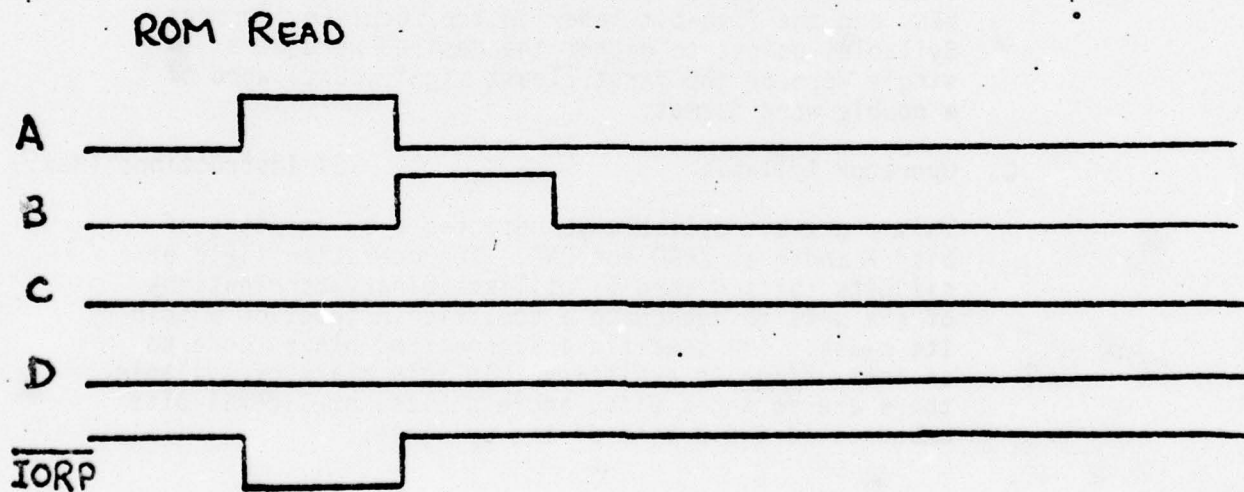
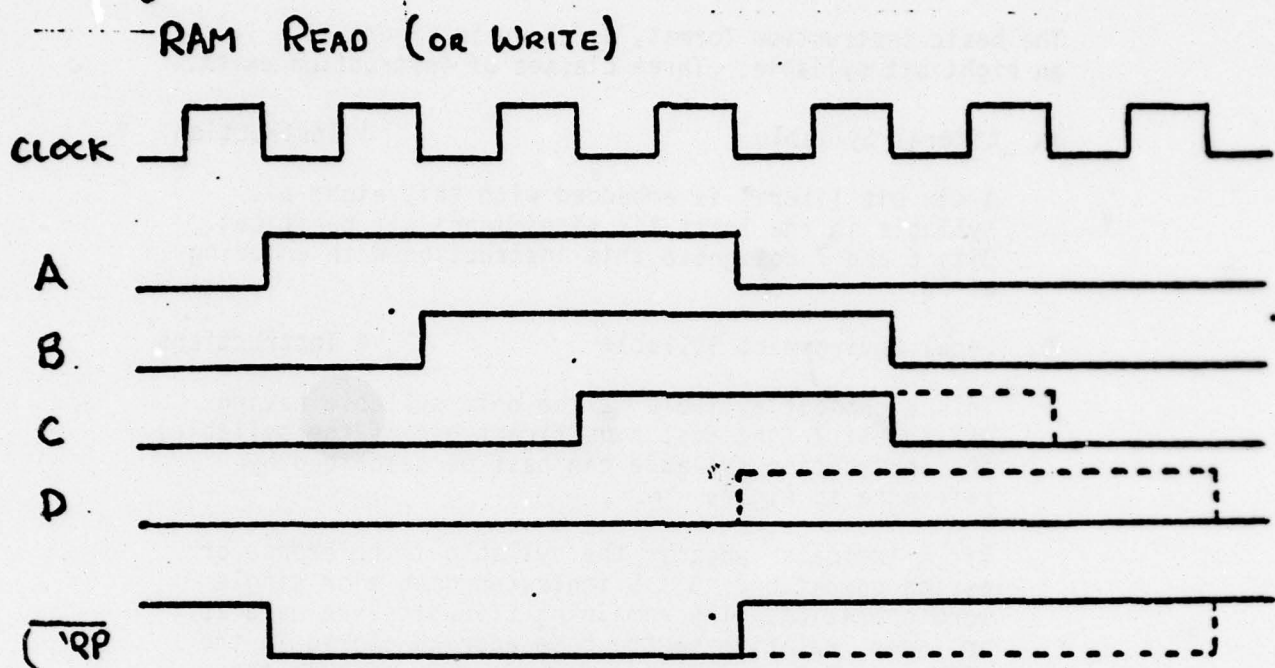


Figure 25. Memory Control Timing

2.5.3.6 INSTRUCTION SET

2.5.3.6.1 INSTRUCTION FORMAT

The basic instruction format, as shown in Figure 26 is an eight bit syllable. Three classes of instruction exist.

A. Literal Syllable

1 Instruction

A six bit literal is embedded with this eight-bit syllable in the least six significant bit positions. Bits 6 and 7 designate this instruction with encoding of 00.

B. Local Environment Syllable

4 Instructions

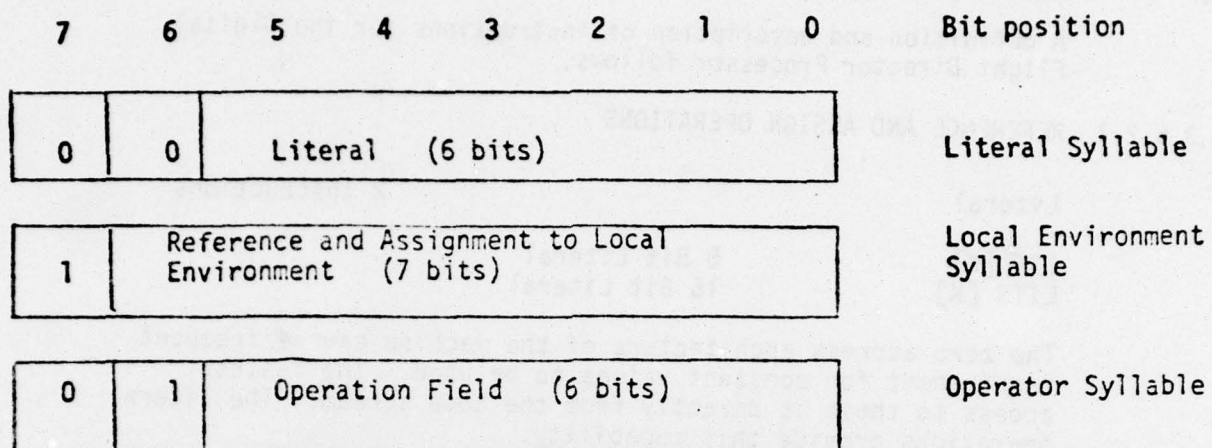
This eight-bit syllable is the only syllable having ONE in bit 7 (the most significant bit of the syllable). The instruction syllable can best be described by reference to Figure 26.

Bit 6 indicates whether the syllable is reference or assign operation. Bit 5 indicates double or single word operation. The remaining five bits are used as an index, relative to the base address stored in the LENV (Local Environment Pointer), of a word in the local environment. The computed address (addition of LENV and the five-bit index in the Local Environment Syllable) points to either the desired word of a single word or the first (least significant) word of a double word format.

C. Operator Syllable

64 Instructions (Max.)

This eight-bit syllable is detectable by encoding of bits 7 and 6 as ZERO and ONE. The operation field of six bits (bits 0 thru 5) utilizes binary combinations of six bits to designate a specific instruction within its class. The specific assignment of binary code to an instruction is arbitrary. In this class of syllable there are no index bits, address bits, or literal bits embedded in the 8 bits of the syllable.



INSTRUCTION FORMAT

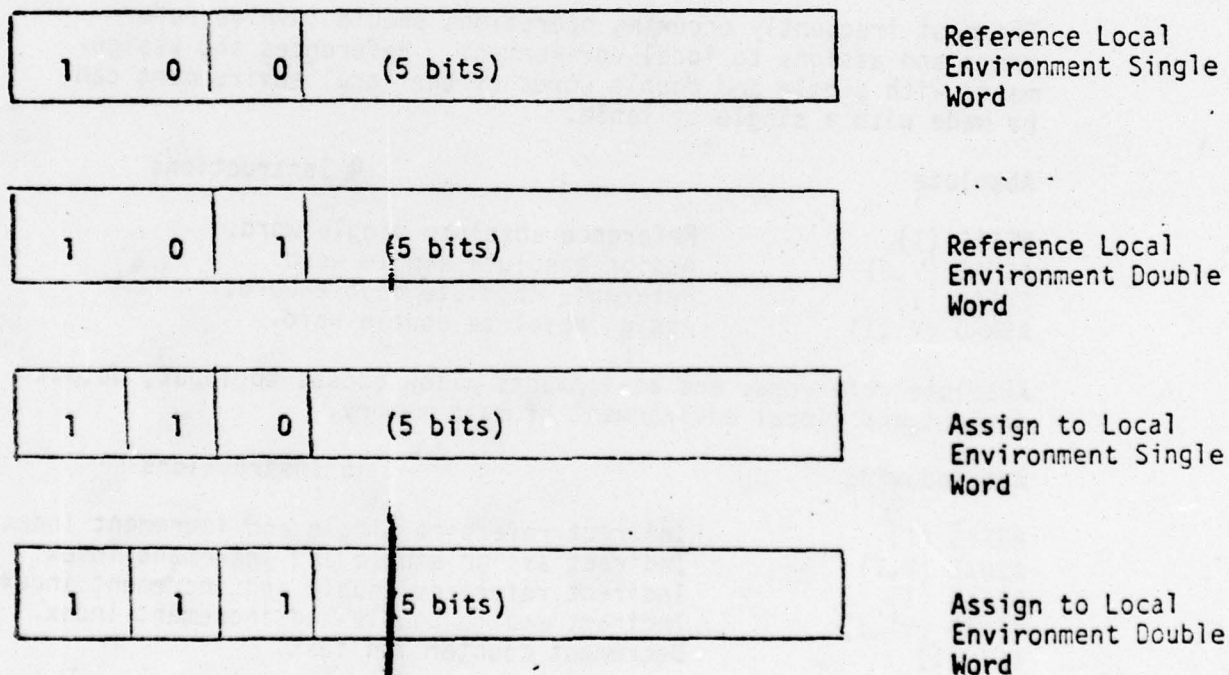


Figure 26. Format of Local Environment Syllable

2.5.3.6.2 INSTRUCTION DEFINITION

A definition and description of instructions for the Digital Flight Director Processor follows.

2.5.3.6.2.1 REFERENCE AND ASSIGN OPERATIONS

Literal

2 Instructions

LIT [K]	6 Bit Literal
LITS [K]	16 Bit Literal

The zero address architecture of the machine causes frequent requirement for constant values to be used. The fastest access to these is directly from the code stream. The literal operations provide this capability.

Local Environment Relative

5 Instructions

LOCL (I)	Global relative address of local word.
REFLS [K]	Reference local environment single word.
ASNLS ([K],V)	Assign to local environment single word.
REFLD [K]	Reference local environment double word.
ANSLO ([K],V')	Assign to local environment double word.

The most frequently occurring operations should involve references and assigns to local environment. References and assignments with single and double words of the local environment can be made with a single syllable.

Absolute

4 Instructions

REFAS (I)	Reference absolute single word.
ASNAS (V,I)	Assign absolute single word.
REFAD (I)	Reference absolute double word.
ASNAD (V',I)	Assign absolute double word.

Absolute references and assignments allow access to input, output devices and global environment of main memory.

Autoindexing

5 Instructions

REFXS (I)	Indirect reference single and increment index.
ASNXS (V,I)	Indirect assign single and increment index.
REFXD (I)	Indirect reference double and increment index.
ASNXD (V',I)	Indirect assign double and increment index.
DCT (I)	Decrement counter and test.

These instructions are used for indirect accessing and storage of single and double word quantities indirectly through index registers. A decrement and test instruction provides a counting capability with iterative control.

Stack

2 Instructions

DUPS (V)	Duplicate top of the stack single word.
DUPD (V')	Duplicate top of the stack double word.

2.5.3.6.2.2 These stack manipulation operations are useful for nondestructive assignments and other machine level macros.

CONTROL OPERATIONS

Transfer

3 Instructions

SKIP (N)	Unconditional skip N syllables.
IFT (C,N)	If condition TRUE, continue, else skip.
IFF (C,N)	If condition FALSE, continue, else skip.

The control transfer operators unconditionally or conditionally alter the contents of the program syllable count register (SPCR). They are designed so that all labels appear as SPCR relative addresses in the code.

Subroutine and Block

4 Instructions

CALL (X1.....XN,N,F)	Call subroutine.
RETURN	Return from subroutine.
BEGIN (X1....XN)	Block begin.
END	Block end.

The subroutines make use of process stack to store their actual parameters, return values, and local variables. Consequently recursive routines are primitive to this machine. The BEGIN instruction places an arbitrary number of values in the accumulator region of stack into the local environment region of the stack. The END instruction complements BEGIN instruction.

Loop

3 Instructions

SVSKIP (N)	Save current SPCR and skip N-syllable.
LOOP (L,C)	Loop on condition true.
LOOPF (L,C)	Loop on condition false.

The iterative control operations provide several specialized forms of loop control. Instruction SVSKIP can be used with control transfer operations to form other iterative controls.

Interrupt

1 Instruction

INTRTN

Interrupt return (executed in interrupt mode)

HALT

Interrupt mode entry (executed in user mode)

The function of the INTRTN/HALT instruction is dependent upon the mode (interrupt mode or user mode) at the time of execution. If executed while the machine is in the interrupt mode, the function is that of interrupt return; that is, the return of control state to some user mode program. If the instruction is executed while the machine is in user mode, the HALT function allows suspension of the current user mode program and mode transfer to the interrupt mode.

2.5.3.6.2.3

ARITHMETIC OPERATIONS

TWOs complement fractional

4 Instructions

ADD (X,Y)

TWOs complement add.

SUB (X,Y)

TWOs complement subtract.

MPY (X,Y)

TWOs complement multiply.

DIV (X,Y)

TWOs complement divide.

Extended precision TWOs complement fractional

EADD (R,X,Y)

ESUB (R,X,Y)

SMPY (R,X,Y)

EDIV (R,X,Y)

2.5.3.6.2.4

LOGICAL AND RELATIONAL OPERATIONS

Boolean

1 Instruction

BOOL (X,Y,N)

Bit by bit Boolean function N.

Relational

2 Instructions

GR (X,Y)

TWOs complement fraction greater than.

EQ (X,Y)

TWOs complement fraction equal.

Test and Set

3 Instructions

SET (S,I)

Test and set bit.

RESET (S,I)

Test and reset bit.

TRIB (S)

Test and reset interrupt bit.

The SET and RESET operations allow a bit from an absolutely addressed word (including any register) to be explicitly read left in a particular state. The operation TRIB can be used only with the Interrupt Register.

2.5.3.6.2.5 MANIPULATIVE

XTRACT (X,S,L)	Extract field from word.
INSERT (X,Y,X,L)	Insert field in word.

These two instructions permit partial word operations of a continuous group of bits in a word treated as a ring.

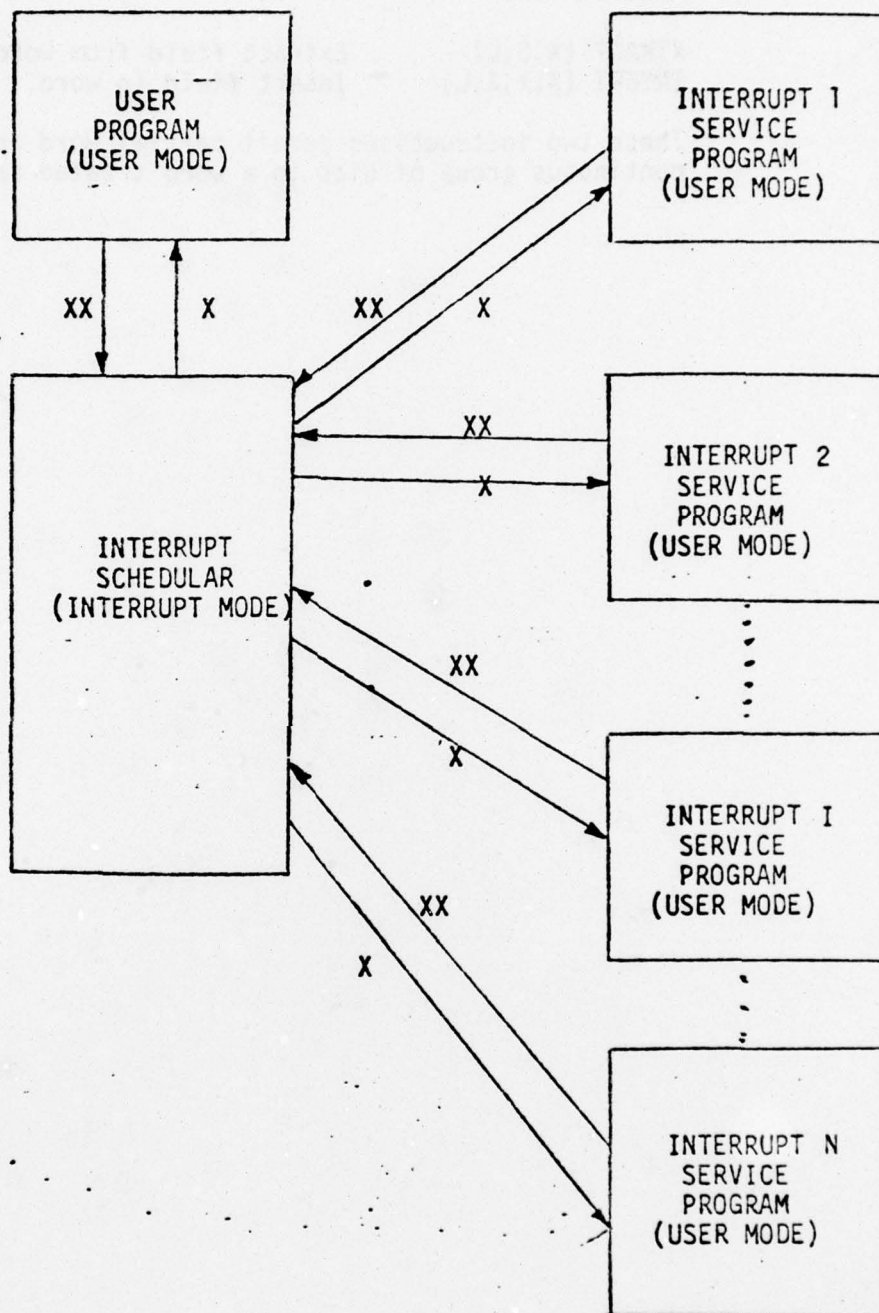


Figure 27. Program Configuration

X INTRTN
XX INTERRUPTS (EXTERNAL OR HALT)

2.5.3.6.3 INTERRUPT HANDLING

There are two modes of processing in the machine, the interrupt mode and user mode. The machine normally operates in user mode. When an interrupt condition is generated, either externally or internally, interrupt processing is indicated. An interrupt mask register is used to indicate which interrupts are recognized.

Figure 27 shows the program configuration in the machine. Each program is described by its own code descriptor and stack descriptor. Control can be transferred from one program to the other via interrupt and interrupt scheduler program. Queuing of interrupts is done in the interrupt scheduler program, in the interrupt mode. The interrupts are serviced in the user mode. Since the interrupt scheduler program is executed frequently, it is desirable that the interrupt scheduler be a short program.

2.5.3.6.3.1 Sequence of Interrupt Handling

The sequence of interrupt handling can best be described by reference to Figure 28.

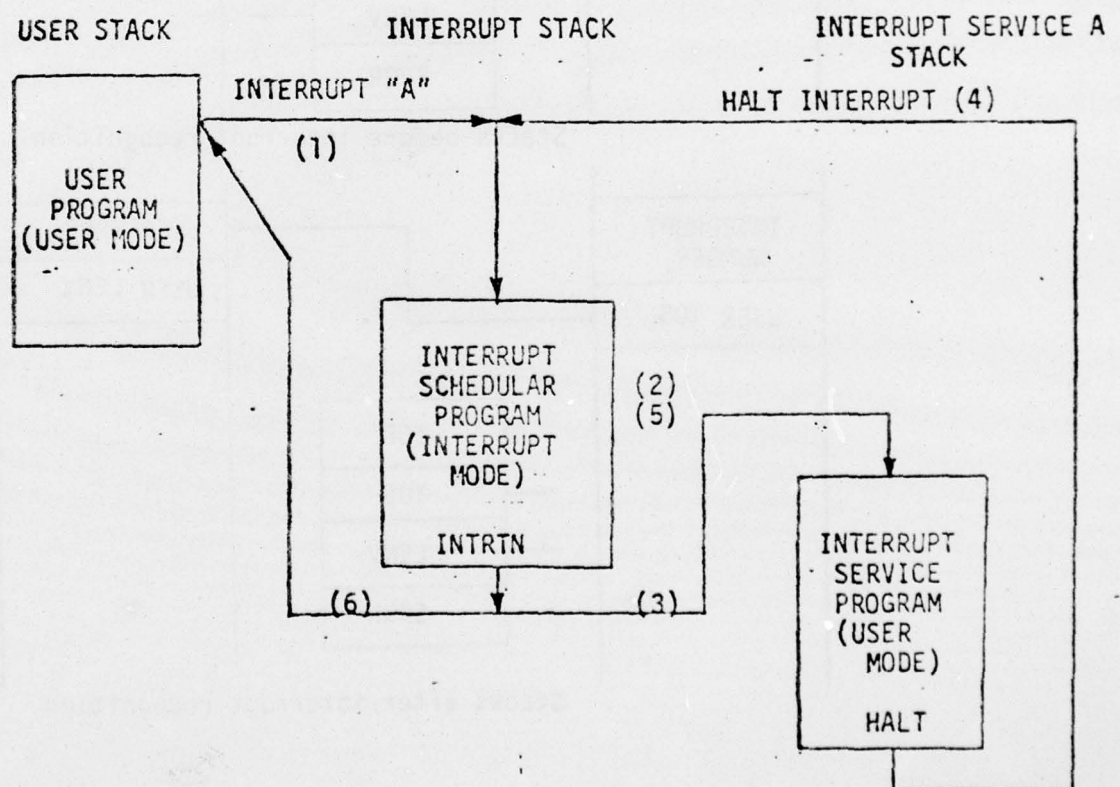


Figure 28. Interrupt Handling Sequence

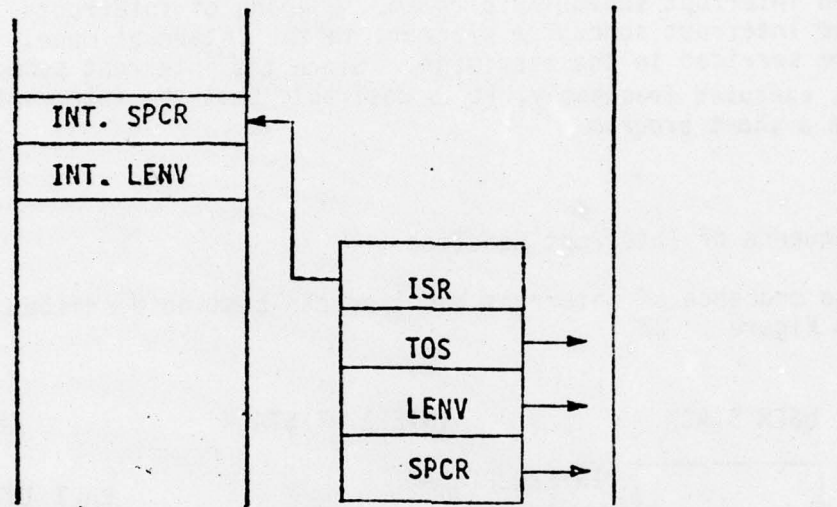
Notes: (1) User stack is active when interrupt A is recognized

Recognition of interrupt activates the interrupt stack, and user stack is suspended.

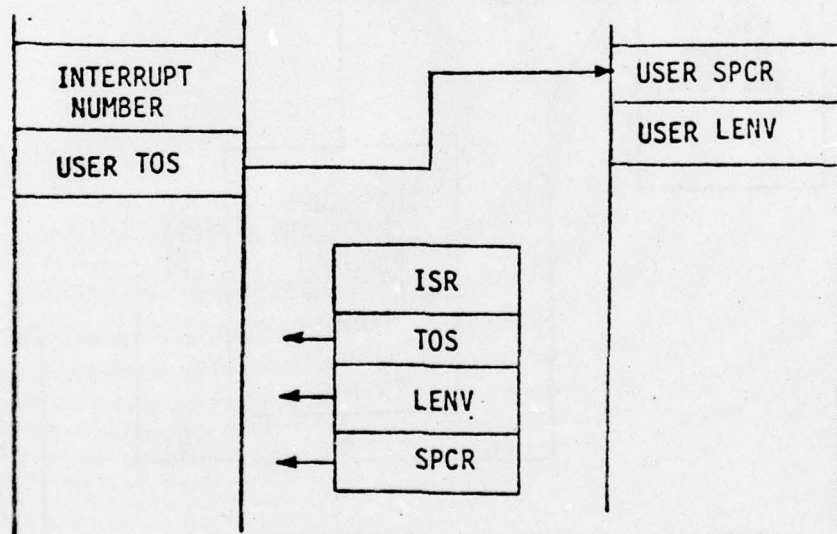
Stacks are shown before and after recognition of interrupts

Interrupt Stack

User Stack



Stacks before interrupt recognition

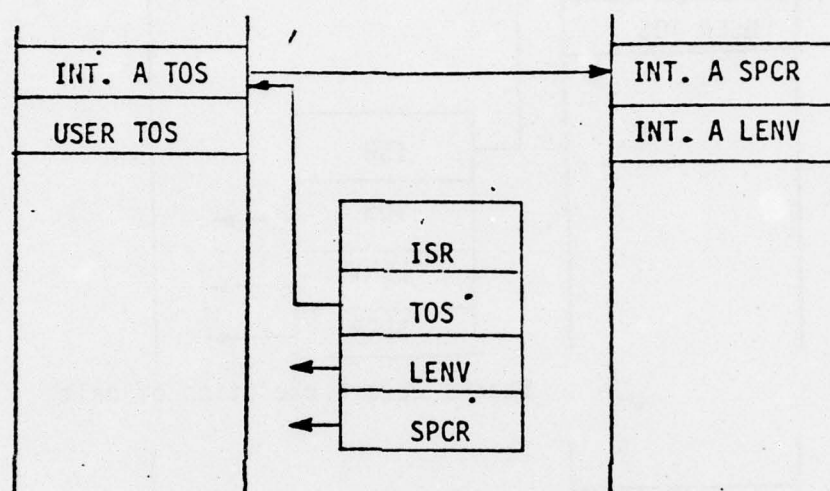


Stacks after interrupt recognition

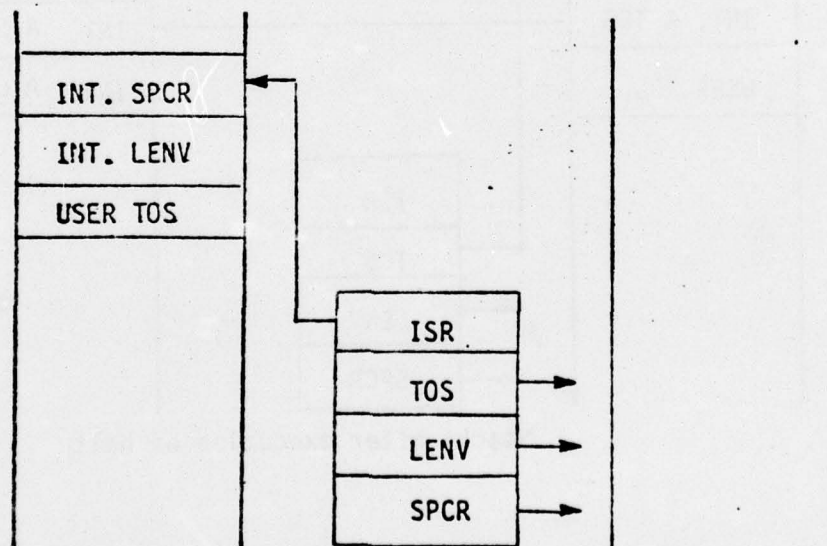
- (2) Interrupt scheduler program identifies and processes the received interrupt.
- (3) Execution of INTRTN instruction transfers control to interrupt A service program. Interrupt stack is suspended and interrupt A service stack is activated.

Interrupt Stack

Interrupt A Service Stack



Stacks before execution of INTRTN

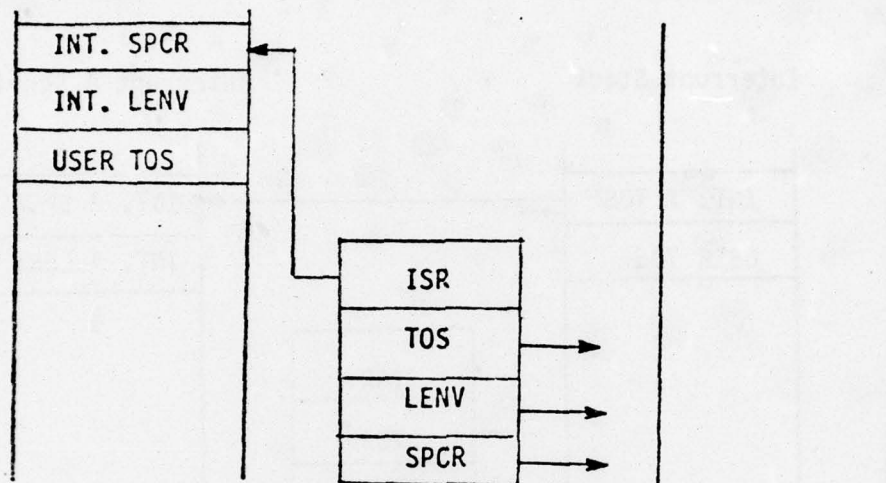


Stacks after execution of INTRTN

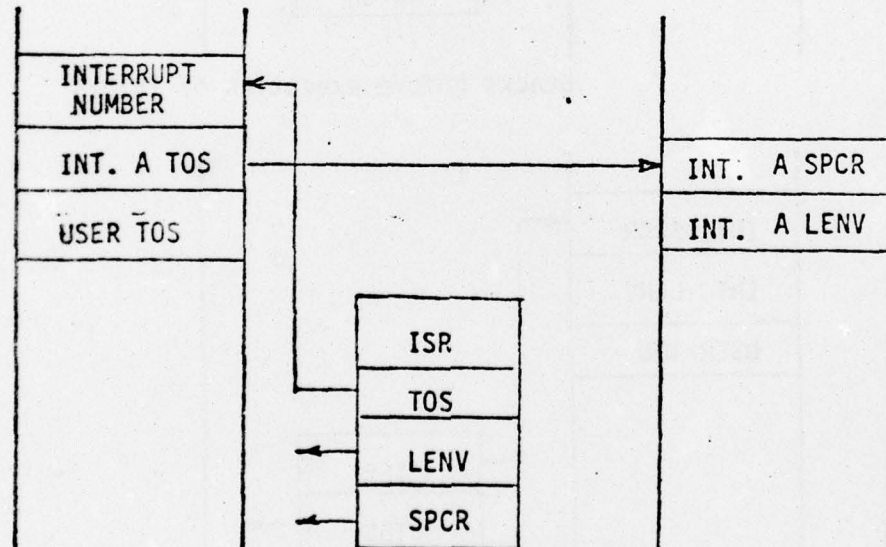
- (4) After interrupt A service program is complete halt instruction is executed. This generates halt interrupt. The control is transferred to interrupt scheduler program. 'Interrupt A' service stack is suspended.

Interrupt Stack

Interrupt A Service Stack



Stacks before execution of halt

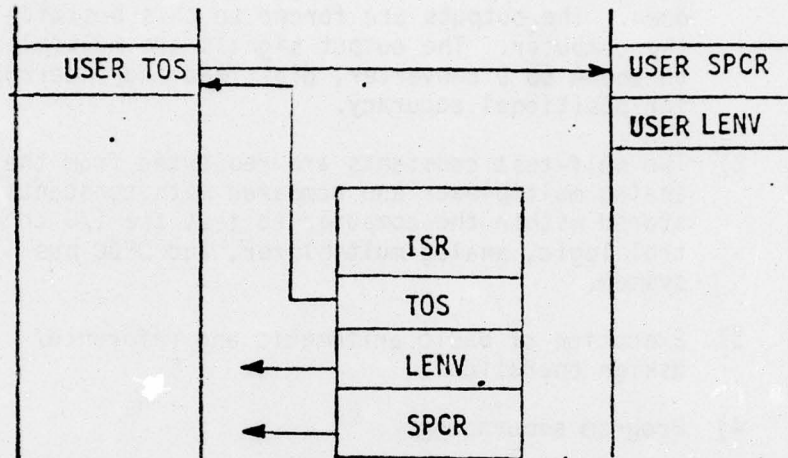


Stacks after execution of halt

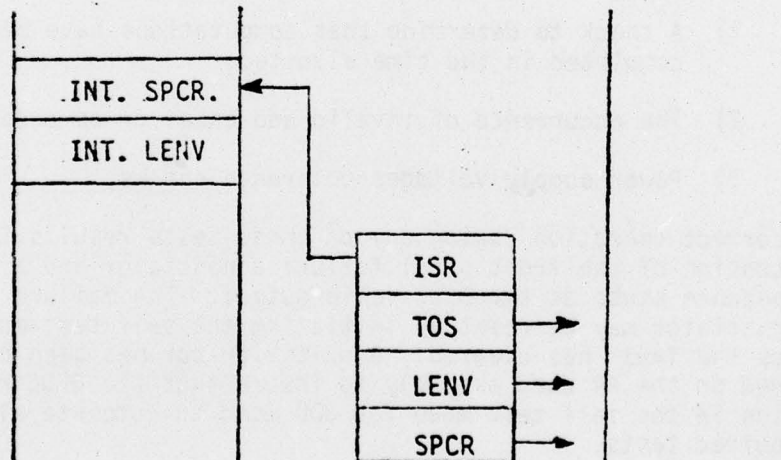
- (5) Halt interrupt is processed.
- (6) Execution of user program is resumed. Interrupt stack is suspended.

Interrupt Stack

User Stack



Stacks before execution of INTERTN



Stacks after execution of INTRTN

2.6

Self Test

Initiation of the Self Test Mode, either by the external 28 volt discrete signal or the front panel pushbutton causes the DFDC to perform internal tests which evaluate the integrity of the unit. These tests are:

- 1) Deflection of the Vertical and Horizontal pointers one quarter inch to the right and one quarter inch down. The outputs are forced to this position by the computer. The output signals are multiplexed to the A to D converter, digitized and interrogated for positional accuracy.
- 2) Two self-test constants are requested from the analog multiplexer and compared with constants stored within the computer to test the I/O control logic, analog multiplexer, and DFDC bus system.
- 3) Execution of basic arithmetic and reference/assign operations.
- 4) Program sequencing.

When the DFDC is not in the self test mode certain monitor functions are performed:

- 1) A check to determine that computations have been completed in the time allotted.
- 2) The occurrence of invalid addresses or op codes.
- 3) Power supply voltages tolerance checks.

Incorrect operation during any of these tests results in actuation of the front panel failure annunciator and a high impedance state at the DFDC valid output. The failure annunciator may be reset by initiating the self test mode once the fault has cleared. A multivibrator has been provided on the A4 card assembly to insure that the DFDC remains in the self test mode for 300 msec to complete all required tests.

2.7

Power Requirements

The DFDC operates from single phase 115V 400 HZ aircraft power. Power supply design ensures operation from 105 to 125V and 320 to 480 HZ. The unit dissipates 53.2 watts distributed as follows:

Card Assembly	Power (Watts)
A1	.005
A2	.425
A3	1.515
A4	3.721
A5	1.390
A6	1.375
A7	3.610
A8	3.395
A9	3.800
A10	3.675
A11	4.070
<hr/>	
Card Total	27.081
Power Supply	26.12

2.8

Power Converter

Three series regulators implemented with monolithic devices provide +15, -15, and -12V. Each supply will continuously output 40% more current than is required for the Digital Flight Director. The addition of short circuit protection and foldback current limiting insures that these supplies will survive under the most adverse operating conditions.

A switching regulator provides +5 volt logic power. This unit normally operates at 4.5 amps and is capable of supplying 7.5 amps continuously. Short circuit protection is implemented by limiting the series switch duty cycle to 90%.

Power supply design specifications are listed below:

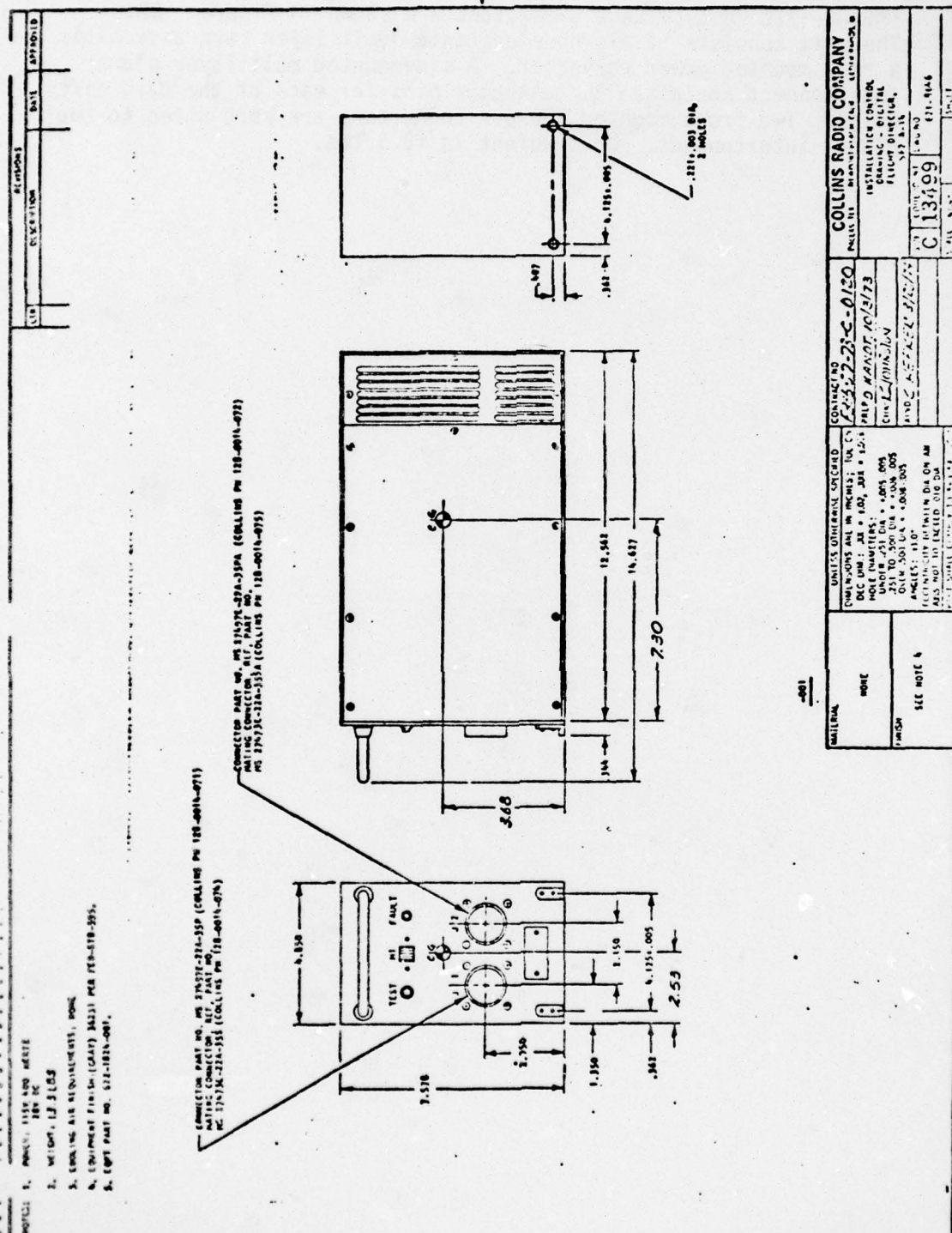
Output Current

	<u>Normal Operation</u>	<u>Maximum Capability</u>	<u>Regulation</u>
+5V	4.5 amps	7.5 amps	2%
+15	180 ma	250 ma	1%
-15	100 ma	250 ma	1%
-12	135 ma	500 ma	1%
-5	40 ma	100 ma	2%

2.9

Configuration

The DFDC is contained in an ARINC one-half ATR short rack (MIL-C-172C case). Case dimensions are shown in Figure 29. The unit consists of eleven electronic multilayer card assemblies and a rear mounted power converter. A sidemounted multilayer planar interconnect contains 130 connector pins for each of the card assemblies. Two front mounted 100 pin connectors are hard wired to the planar interconnect. Unit weight is 13.3 lbs.



1. PARTS LIST AND SEE 2. WEIGHT, 12.3 LBS 3. EQUIPMENT AIR REQUIREMENTS, NONE 4. EQUIPMENT FINISH (GRAT) 3033 PER FED-STD-395 5. EQUIPMENT NO. 522-1821-001		1. PARTS LIST AND SEE 2. WEIGHT, 12.3 LBS 3. EQUIPMENT AIR REQUIREMENTS, NONE 4. EQUIPMENT FINISH (GRAT) 3033 PER FED-STD-395 5. EQUIPMENT NO. 522-1821-001	
1. PARTS LIST AND SEE 2. WEIGHT, 12.3 LBS 3. EQUIPMENT AIR REQUIREMENTS, NONE 4. EQUIPMENT FINISH (GRAT) 3033 PER FED-STD-395 5. EQUIPMENT NO. 522-1821-001		1. PARTS LIST AND SEE 2. WEIGHT, 12.3 LBS 3. EQUIPMENT AIR REQUIREMENTS, NONE 4. EQUIPMENT FINISH (GRAT) 3033 PER FED-STD-395 5. EQUIPMENT NO. 522-1821-001	

Figure 29. DFDC Installation Control Drawing

AD-A041 626

ROCKWELL INTERNATIONAL CEDAR RAPIDS IOWA COLLINS RA--ETC F/G 17/7
DIGITAL FLIGHT DIRECTOR COMPUTER.(U)
APR 75 J P DESMOND, G E FORQUER

UNCLASSIFIED

ASD-TR-76-3

F33657-73-C-0120
NL

2 of 4

ADA041626

1



2.10 Airspeed Controller

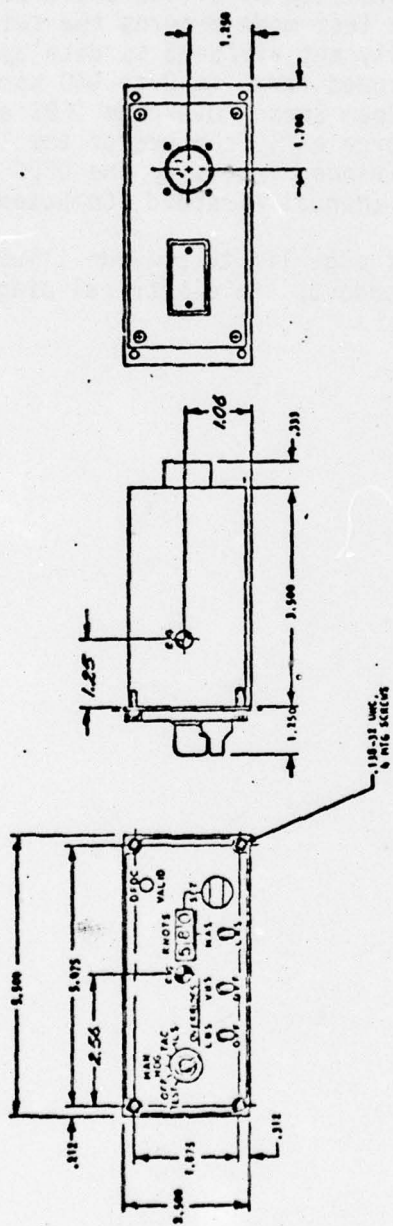
The 377B-1 Air Speed Controller (ASC) provides mode control, airspeed input data, beam sensor overrides, and airspeed input selection to the DFDC. Outside dimensions of the ASC are illustrated in Figure 30.

DFDC mode control is provided by a five position rotary switch. A spring return in the test mode returns the switch to the OFF mode when released. Manually set airspeed is displayed on three 10 digit rotary displays. Airspeed range is 0 to 840 knots. By setting the Lateral and Vertical Beam Sensor Override (LBS and VBS) toggle switches to ON the pilot can force early capture of the localizer and glide-slope radio beams. Airspeed inputs to the DFDC are selected by the MAS CAS toggle switch (Manual Airspeed, Computed Airspeed).

The ASC front panel is edge lit to provide illumination of switch labels and airspeed readout. An electrical diagram of the ASC is shown in Figure 31.

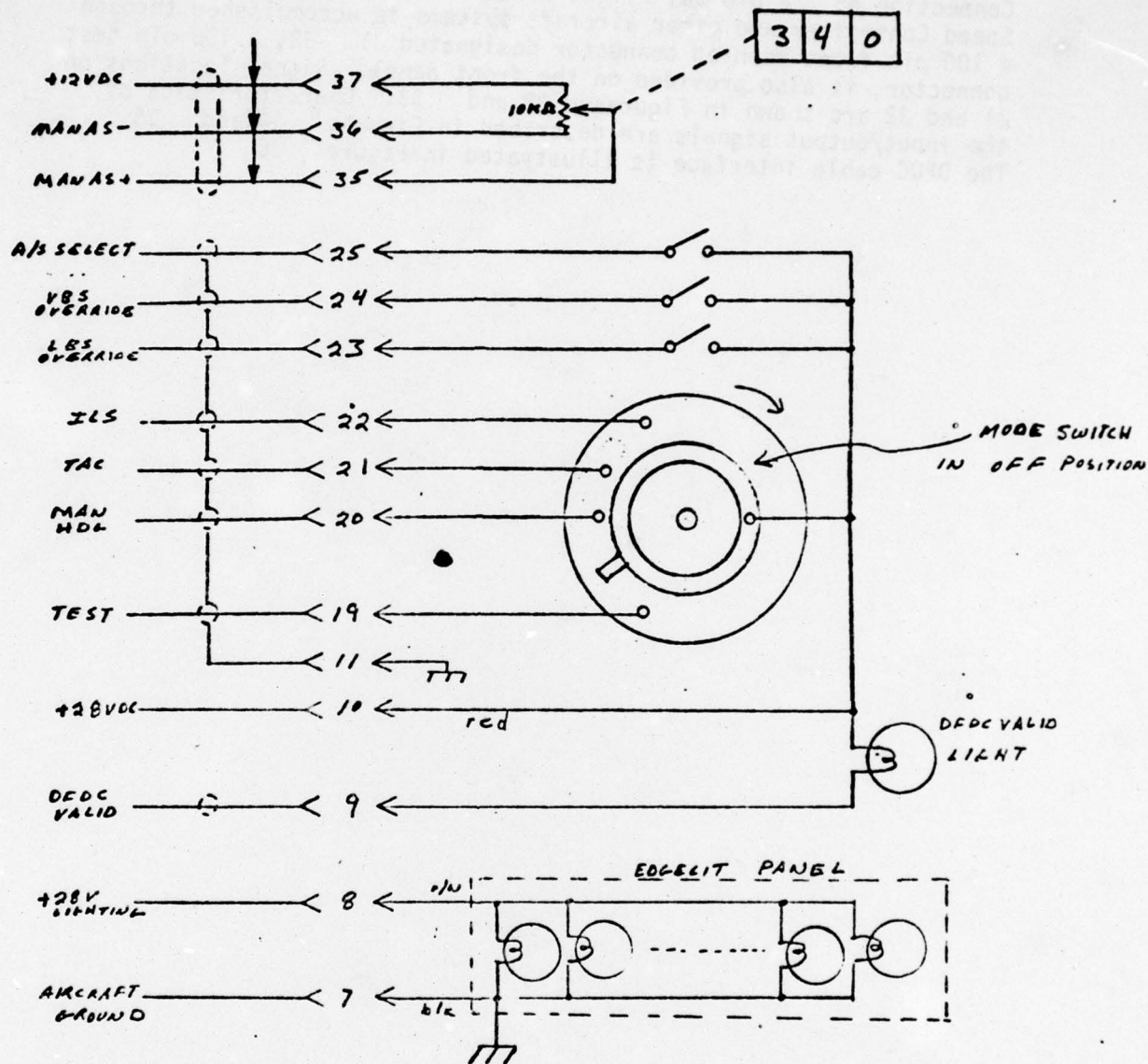
REV.	DESCRIPTION	DATE	APPROVED
1			

1. ELECTRICIAN, COMBINATION (A), INSTRUMENT-11A-355, NOTES WITH INSTRUMENT-11A-355.
 2. INSTRUMENT-11A-355, NOTES WITH INSTRUMENT-11A-355.
 3. INSTRUMENT-11A-355, NOTES WITH INSTRUMENT-11A-355.
 4. INSTRUMENT-11A-355, NOTES WITH INSTRUMENT-11A-355.
 5. INSTRUMENT-11A-355, NOTES WITH INSTRUMENT-11A-355.



MATERIAL	COLLINS RADIO COMPANY	
	BELLUS 11A-355	
SPECIFICATION	INSTALLATION CONTROL DRAWING	
	CONTROLLER, AIR SPEED, 3770-1	
SEE NOTE 4	C13029	
	C13029	

Figure 30. ASC Installation Control Drawing



AIRSPPEED CONTROLLER

377B-1

C/N 622-1825-001

Figure 31. ASC Electrical Diagram

2.11 Aircraft Interconnect

Connection of the Digital Flight Director Computer to the Air Speed Controller and other aircraft systems is accomplished through a 100 pin front mounted connector designated J1. J2, a 100 pin test connector, is also provided on the front panel. Signal locations on J1 and J2 are shown in Figures 32 and 33. Characteristics of the input/output signals are described in Figures 34 and 35. The DFDC cable interface is illustrated in Figure 36.

<u>PIN</u>	<u>SIGNAL</u>	<u>PIN</u>	<u>SIGNAL</u>
1	SPARE	51	COMPUTED AIR SPEED +
2	SPARE	52	COMPUTED AIR SPEED -
3	SPARE	53	MANUAL AIR SPEED +
4	SPARE	54	MANUAL AIR SPEED -
5	SPARE	55	+12 VDC
6	SPARE	56	SPARE
7	SPARE	57	PITCH Y
8	SPARE	58	PITCH X
9	HEADING ERROR H	59	PITCH Z
10	HEADING ERROR C	60	HORIZONTAL POINTER +
11	LATERAL (HSI) DEV FLAG +	61	HORIZONTAL POINTER -
12	LATERAL (HSI) DEV FLAG -	62	ILS MODE
13	GLIDESLOPE FLAG +	63	TAC MODE
14	GLIDESLOPE FLAG -	64	RADAR ALTIMETER VALID
15	SPARE	65	RADAR ALTIMETER COMMON
16	SPARE	66	POWER COMMON
17	ERROR REFERENCE H	67	SPARE
18	ERROR REFERENCE C	68	LOCALIZER DEVIATION +
19	VERTICAL POINTER FLAG	69	LOCALIZER DEVIATION -
20	HORIZONTAL POINTER FLAG	70	LATERAL (HSI) DEV OUTPUT +
21	GLIDE PATH ENGAGE	71	LATERAL (HSI) DEV OUTPUT -
22	FLIGHT DIRECTOR VALID	72	MANUAL HEADING
23	SELF TEST	73	RADAR ALTITUDE +
24	+28 VDC	74	RADAR ALTITUDE -
25	SPARE	75	LOCALIZER FLAG -
26	COURSE ERROR H	76	SPARE
27	COURSE ERROR C	77	SPARE
28	DME HUNDREDS X	78	TAC DEVIATION +
29	DME HUNDREDS Y	79	TAC DEVIATION -
30	SPARE	80	VERTICAL POINTER +
31	VBS OVERRIDE	81	VERTICAL POINTER -
32	LBS OVERRIDE	82	TAC FLAG +
33	SPARE	83	TAC FLAG -
34	SPARE	84	LOCALIZER FLAG +
35	SPARE	85	SPARE
36	DME UNITS X	86	SPARE
37	DME UNITS Y	87	GLIDESLOPE DEV -
38	DME TENS X	88	GLIDESLOPE DEV +
39	DME TENS Y	89	GLIDESLOPE DEV OUTPUT +
40	SPARE	90	GLIDESLOPE DEV OUTPUT -
41	AIR SPEED SELECT	91	GLIDESLOPE FLAG OUT +
42	AIR SPEED REFERENCE +	92	GLIDESLOPE FLAG OUT -
43	AIR SPEED REFERENCE -	93	ADI COMMON
44	SPARE	94	SHIELD GROUND
45	115 VOLTS 400 HZ	95	SPARE
46	ROLL (BANK) X	96	SPARE
47	ROLL (BANK) Y	97	SPARE
48	ROLL (BANK) Z	98	SPARE
49	DME REFERENCE H	99	SPARE
50	DME REFERENCE C	100	CHASSIS GROUND

Figure 32. J1 - Signal Connector

<u>PIN</u>	<u>SIGNAL</u>	<u>PIN</u>	<u>SIGNAL</u>
1	SPARE	51	DB01F
2	SPARE	52	DB02F
3	SPARE	53	DB03F
4	SPARE	54	DB04F
5	SPARE	55	DB05F
6	PCOS	56	DB06F
7	PSIN	57	DB07F
8	RCOS	58	DB08F
9	RSIN	59	DB09F
10	RHSIN	60	DB10F
11	RHCOS	61	DB11F
12	RTSIN	62	DB12F
13	RT COS	63	DB13F
14	GS	64	DB14F
15	LOC	65	DB15F
16	TAC	66	AB00F
17	RALT	67	AB01F
18	CAS	68	AB02F
19	GS FLG	69	AB03F
20	LOC FLG	70	AB04F
21	TAC FLG	71	AB05F
22	REFAS	72	AB06F
23	EOC	73	AB07F
24	THETAHE	74	AB08F
25	THETACE	75	AB09F
26	OFFF	76	AB10F
27	ILSF	77	AB11F
28	FIRESET	78	AB12F
29	FISET	79	BRQF
30	STC1	80	SZBF
31	STC2	81	BG0
32	CLK8M	82	BG1
33	POC	83	BG2
34	PWROUTF	84	IORWF
35	CLK.5M	85	MINT
36	CLK.25M	86	IORQF
37	T1	87	IORPF
38	T2	88	SPARE
29	IOINT	89	DEVENF
40	SROE	90	SPARE
41	ODA	91	SPARE
42	CONVI	92	UM
43	SAH	93	DMARQ
44	LDIS	94	INT8
45	P5V	95	SPARE
46	N5V	96	REFRSF
47	P15V	97	SPARE
48	N15V	98	GND
49	N12V	99	SPARE
50	DB00F	100	SGND

Figure 33. J2 - Test Connector

DIGITAL FLIGHT DIRECTOR INPUT SIGNAL INTERFACE CHARACTERISTICS

SIGNAL	PINS	INTERFACING SYSTEM	SIGNAL CHARACTERISTICS	APPLICABLE SPECIFICATION
1 HEADING ERROR	J1-9 H J1-10 C	HSI	0-11 VAC Referenced to 3	MIL I 83034A Figure 4
2 COURSE ERROR	J1-26 H J1-27 C	HSI	0-11 VAC Referenced to 3	MIL I 83034A Figure 4
3 ERROR REFERENCE	J1-17 H J1-18 C	HSI	26 VAC	MIL I 83034A Figure 4
4 ROLL (BANK)	J1-46 X J1-47 Y J1-48 Z	Roll Gyro	3 wire synchro	MIL C 83014A Figure 4
5 PITCH	J1-57 X J1-58 Y J1-59 Z	Pitch Gyro	3 wire synchro	
6 DME HUNDREDS	J1-28 X J1-29 Y	DME Receiver or HSI	3 wire synchro Z leg connected to common J1-50	MIL I 83034A Para. 3.23.1 Figure 4
7 DME TENS	J1-38 X J1-39 Y	DME Receiver or HSI	3 wire synchro Z leg connected to common J1-50	MIL I 83034A Para. 3.23.1 Figure 4
8 DME UNITS	J1-36 X J1-37 Y	DME Receiver or HSI	3 wire synchro Z leg connected to common J1-50	MIL I 83034A Para. 3.23.1 Figure 4
9 DME REFERENCE	J1-49 H J1-50 C	DME Receiver or HSI	Synchro reference	MIL I 83034A Para. 3.23.1 Figure 4
10 LOCALIZER DEVIATION	J1-68 + J1-69 -	ILS Receiver	$+150\mu a = +2^\circ$ $Z_{in} = 1K\Omega$	MILC 83014A
11 GLIDESLOPE DEVIATION	J1-87 + J1-88 -	ILS Receiver	$+150\mu a = +0.7^\circ$ $Z_{in} = 1K\Omega$	MIL C 83014A
12 TACAN DEVIATION	J1-73 + J1-79 -	TACAN Receiver	$+150\mu a = +10^\circ$ $Z_{in} = 1K\Omega$	MIL C 83014A
13 RADAR ALTITUDE	J1-74 + J1-73 -		-40 VDC/1000 ft	
14 COMPUTED AIR SPEED	J1-51 + J1-52 -	AIR DATA COMPUTER	12 VDC/840 kts	Statement of Work
15 AIRSPEED REFERENCE	J1-42 + J1-43 -	AIR DATA COMPUTER	12 VDC	Statement of Work

Figure 34. DFDC Input Signal Characteristics
(Page 1 of 2)

SIGNAL	PINS	INTERFACING SYSTEM	SIGNAL CHARACTERISTICS	APPLICABLE SPECIFICATION
16 MANUAL AIRSPEED	J1-53 + J1-54 -	Airspeed Controller	12 VDC/840 kts	Statement of Work
17 POWER POWER COMMON	J1-45 H J1-66 C		115 VAC 400 HZ Single Phase 1.5 VA	MIL STD 704A
18 +28 VDC	J1-24		0.1 VA	MIL STD 704A
19 CASE GROUND	J1-10			
20 SELF TEST	J1-23	Airspeed Controller	28 VDC - ON 0 VDC - OFF	
21 GLIDESLOPE FLAG	J1-13 + J1-14 -	ILS Receiver	180 μ a In View 255 μ a Out of View Zin = 1K Ω	MIL C 83014A
22 LOCALIZER FLAG	J1-84 + J1-75 -	ILS Receiver	180 μ a In View 255 μ a Out of View Zin = 1K Ω	MIL C 83014A
23 TACAN FLAG	J1-82 + J1-83 -	TACAN Receiver	180 μ a In View 255 μ a Out of View Zin = 1K Ω	MIL C 83014A
24 RADAR ALTITUDE VALID	J1-64	Radar Alt Rcvr	0 VDC Invalid 5 VDC Valid	MIL C 83014A
25 VBS OVERRIDE	J1-31	Airspeed Controller	0 VDC Off 28 VDC On	MIL C 83014A
26 LBS OVERRIDE	J1-32	Airspeed Controller	0 VDC Off 28 VDC On	MIL C 83014A
27 AIRSPEED SELECT	J1-41	Airspeed Controller	0 VDC - Comp. AS 28 VDC - Manual AS	
28 MAN HDG MODE	J1-72	Airspeed Controller	0 VDC Off 28 VDC On	
29 TAC MODE	J1-63	Airspeed Controller	0 VDC Off 28 VDC On	
30 ILS MODE	J1-62	Airspeed Controller	0 VDC Off 28 VDC On	

Figure 34. D^WC Input Signal Characteristics
(Page 2 of 2)

**DIGITAL FLIGHT DIRECTOR OUTPUT SIGNALS
INTERFACE CHARACTERISTICS**

SIGNAL	PINS	INTERFACING SYSTEM	SIGNAL CHARACTERISTICS	APPLICABLE SPECIFICATION
1 VERTICAL POINTER	J1-80 + J1-81 -	ADI	+2.2ma → +23° +12ma Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
2 HORIZONTAL POINTER	J1-60 + J1-61 -	ADI	+2.2ma +23° +12ma Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
3 LATERAL DEVIATION	J1-70 + J1-71 -	HSI	MODE SIGNAL OFF TAC DEV TAC TAC DEV ILS LOC DEV	MIL C 83014A
4 GLIDESLOPE DEV OUT	J1-87 + J1-88 -	ADI	+15ua → +0.7° +50Q _{ua} Out of View	MIL C 83014A
5 VERTICAL POINTER FLAG	J1-19	ADI	Q _{ua} - In View 300Q _{ua} - Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
6 HORIZONTAL POINTER FLAG	J1-20	ADI	Q _{ua} - In View 300Q _{ua} - Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
7 GLIDESLOPE FLAG OUT	J1-91 + J1-92 -	ADI	Q _{ua} - In View 300Q _{ua} - Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
8 LATERAL DEV FLAG OUT	J1-11 + J1-12 -	HSI	Q _{ua} - In View 300Q _{ua} - Out of View Z _L = 500Ω → 1KΩ	MIL C 83014A
9 GLIDE PATH ENGAGE	J1-21	ADI	Open - Not Engaged 160ma sink - Engaged	MIL C 83014A
10 DFDC VALID	J1-22	Airspeed Controller	Open - Invalid 160ma sink - Valid	MIL C 83014A

Figure 35. DFDC Output Signal Characteristics

DIGITAL FLIGHT DIRECTOR COMPUTER EXTERNAL WIRING DIAGRAM

1J1-MS27497E-22A-35P
1P1-MS27473E-22A-35S
2J1-MS27473E-14A-35P
2P1-MS27473E-14A-35S

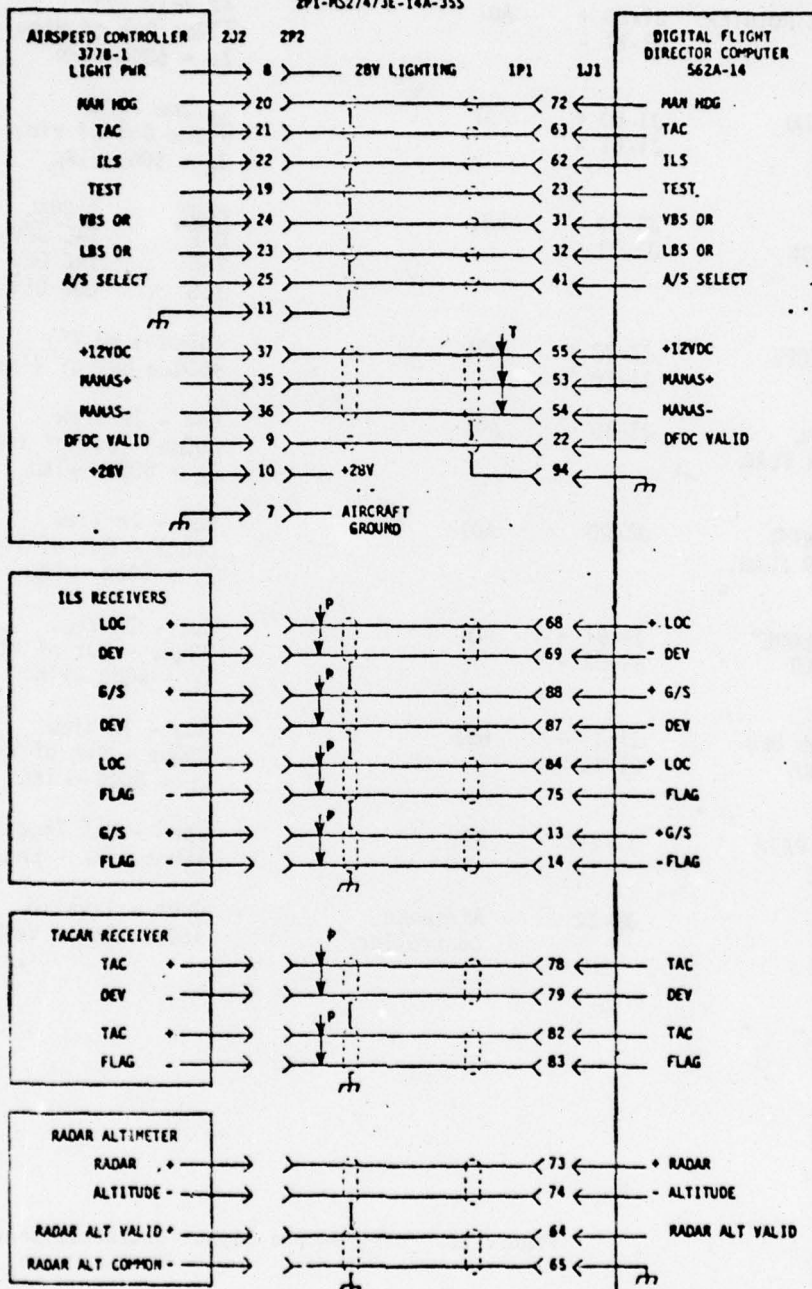


Figure 36. DFDC External Wiring Diagram
(PAGE 1 OF 3)

SCALE
REV
SHEET
B 13499
SIZE
CODE IDENT
ORIG NO

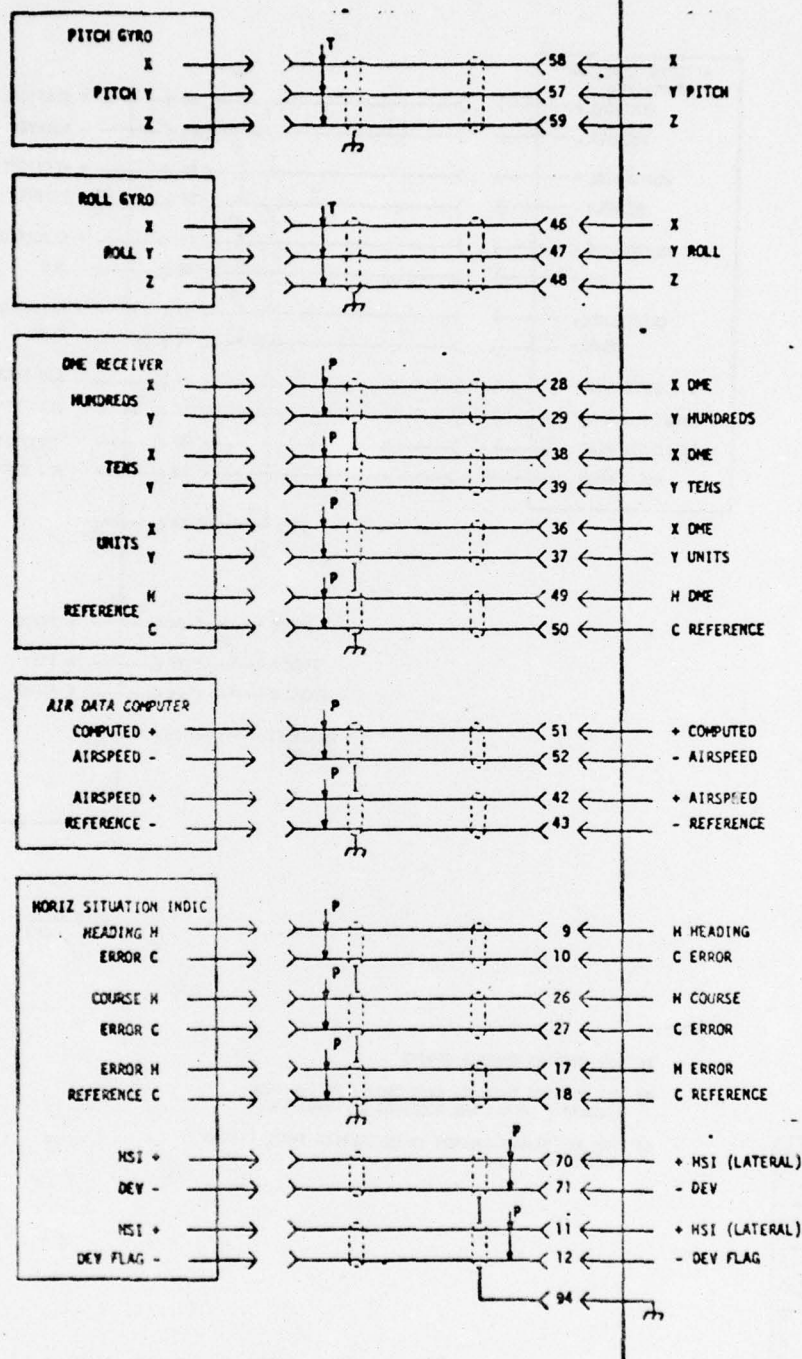
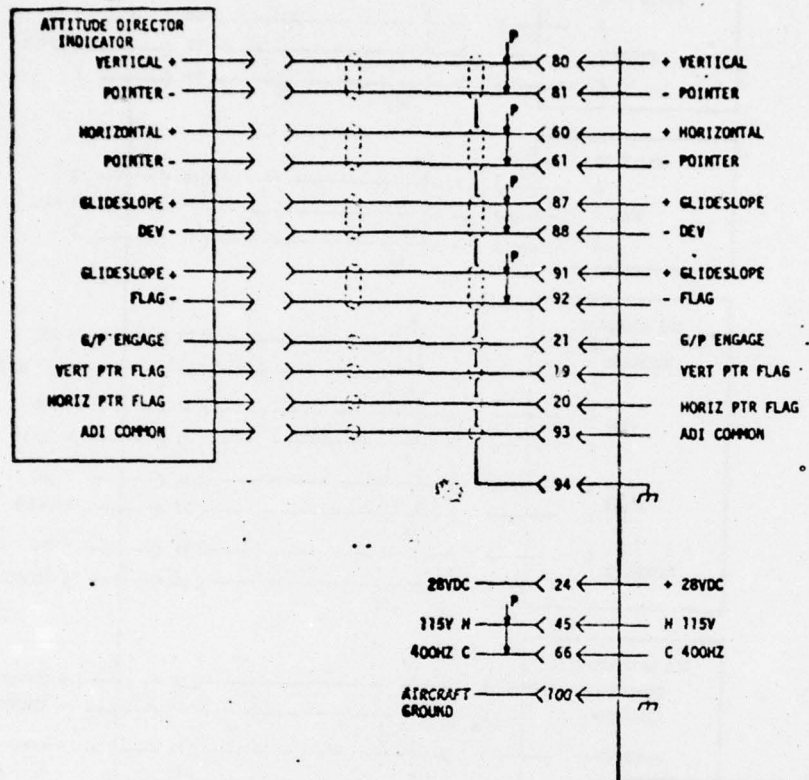


Figure 36. DFDC External Wiring Diagram
(PAGE 2 OF 3)



J. DESMOND 11/6/73
REV 10/7/74

NOTES:

- 1) DENOTES CHASSIS GROUND
- 2) ALL TWISTED SHIELDED PAIR INPUTS TO THE DFDC LABELED + OR H/C ARE RECEIVED DIFFERENTIALLY.
- 3) PIN 94 (SHIELD GROUND) IS REFERENCED THREE PLACES ABOVE.

SCALE	SHEET
B 13499	10W6 H0
REV	

Figure 36. DFDC External Wiring Diagram

(PAGE 3 OF 3)

2.12 Applicability to Mass Production

Prior to hardware design of the prototype computers, guidelines were established to minimize modifications required to produce preproduction units of the 562A-14. To this end, the following items were completed:

- 1) The unit was designed to meet the necessary mechanical, electrical and environmental requirements of MIL-C-83014A.
- 2) A complete set of configuration control drawings and list of materials was prepared.
- 3) Electronic components were selected from off-the-shelf high usage items to insure availability for production quantities.
- 4) Ruggedized, multilayer planar circuit cards were selected for card assemblies and the side plane interconnect.

Although production design and fabrication requirements were imposed on the DFDC, prior to committing the unit in its present form to mass production two alternatives to the present configuration should be considered.

- 1) Processor technology at Collins has advanced to the stage where the present DFDC package size and power requirements will support a processor of three times the DFDC CAPS-2 processor throughput capability (120,000 operations per second). By utilizing Collins state-of-the-art CAPS-4 processor in the present DFDC configuration flight director versatility could be expanded to provide additional mode capability. Attitude hold with pitch synchronization could then be implemented utilizing the present aircraft systems interface. Additional inputs from the Digital Air Data Computer would make possible implementation of altitude, vertical speed, and airspeed hold mode commands through the pitch steering pointer. The CAPS-4 processor is presently being utilized at Collins to perform autoland computations in a dual fail passive autopilot system.
- 2) The second alternative is a reduction of DFDC size power and weight. An investigation of state-of-the-art component advances since the design of the DFDC was undertaken toward this end. Since the processor requires the most space, power and therefore weight this function should be the first candidate for replacement. The possibility of utilizing an mos microprocessor has been considered. However, DFDC processor speed requirements have not yet been achieved by microprocessors.

Basic to the circular capture control law is range estimation which requires a Kalman filter or a filtering technique of comparable complexity. Acceptable performance requires a data update rate of ten per second. These two requirements necessitate a digital processor capable of performing nearly 100,000 operations per second. Present state-of-the-art microprocessor approach 10,000 operations per second - far below what is required.

Until the speed of mos microprocessors is near 100,000 operations per second, it appears the best candidate for a production DFDC would contain a CAPS-4 computer assigned additional tasks.

SECTION III

3.0 Control Law Design and Analysis

Section 3.0 provides detailed analytic results that have evolved during the development and support phases of the Digital Flight Director Computer (DFDC) program. Paragraph 3.1 provides a discussion of the concepts involved in the circular capture algorithm. The circular capture algorithm includes the circular capture of the radio beam to be acquired, the wind estimation, and the estimation of the aircraft ground velocity. This discussion will use variables presented in the originally proposed circular capture algorithm as illustrated in Figure 44. The symbols* will be presented as scripted characters** where a \sim indicates the best estimate of a variable and a $\hat{\sim}$ indicates the predicted value of that variable. The symbols are defined in Table 3.1. The standard aircraft distance and angle definitions are illustrated in Figure 37. The basic geometric relationships for ILS localizer and glideslope are illustrated in Figure 38 and 39, respectively. The algorithm implemented involves three types of variables -- measured, predicted, and estimated. A recognition of the distinction between these is crucial to an understanding of the multistage decision process involved in the control-estimation process. Basically, certain variables can be measured such as $(U_0)_k$, θ_k , ψ_k . Like all real world measurements, these are imperfect either due to corrupting noise or imperfect measuring devices. An example of an estimated variable would be aircraft ground velocity, V_g , a necessary variable in beam capture computations based on geometry. The DFDC measures airspeed and course datum error and in some way estimates aircraft ground velocity by compensating for the effects of wind velocity. The predicted variables are those terms computed from the estimated and measured variables to where the DFDC "thinks" the aircraft will move. The Kalman estimation process which is the principal process by which an estimate of radio beam and beam rate are derived is addressed in Paragraph 3.2. The TACAN control laws which are designed to capture and track TACAN radio information with provisions to minimize overstation problems are presented and discussed in Paragraph 3.3. The localized control laws which are designed to capture and track ILS localizer radio information are presented and discussed in Paragraph 3.4. The glideslope control laws which are designed to capture and track ILS glideslope radio information are presented and discussed in Paragraph 3.5.

3.1 Circular Capture Algorithm

The DFDC was conceived as a general replacement for the CPU-80/A flight director computer but with improved control laws, particularly

* Symbols meaning variables like ϕ , which is bank angle.

**Paragraph 3.1 presents the circular capture algorithm, control laws, and Kalman filter in terms of scripted variables. The remaining sections present terms in the form as they are used in the computer programs. For instance, ϕ would be implemented as PHI, ϕ command would be PHIC, etc.

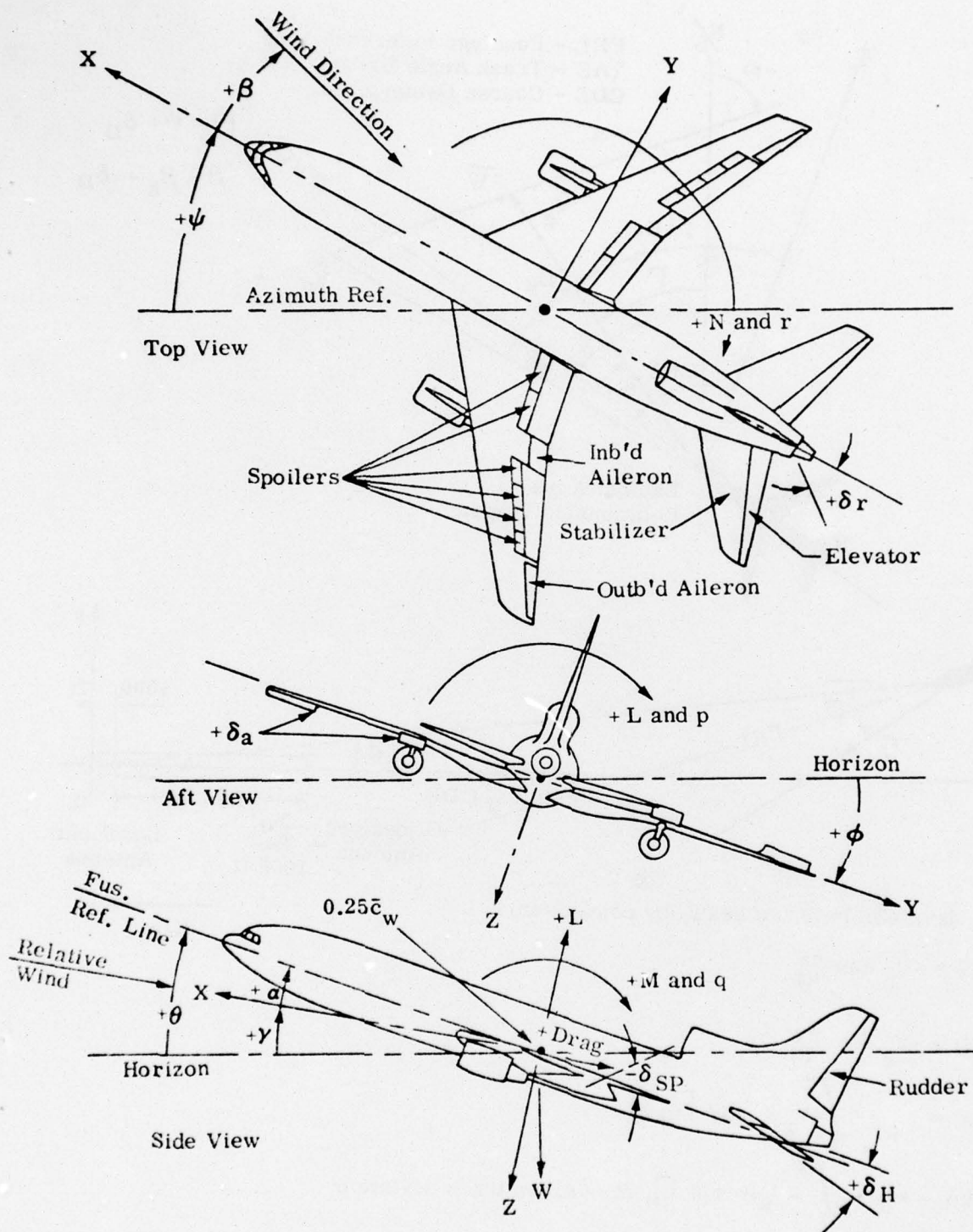


Figure 37. Aircraft Distances and Angles.

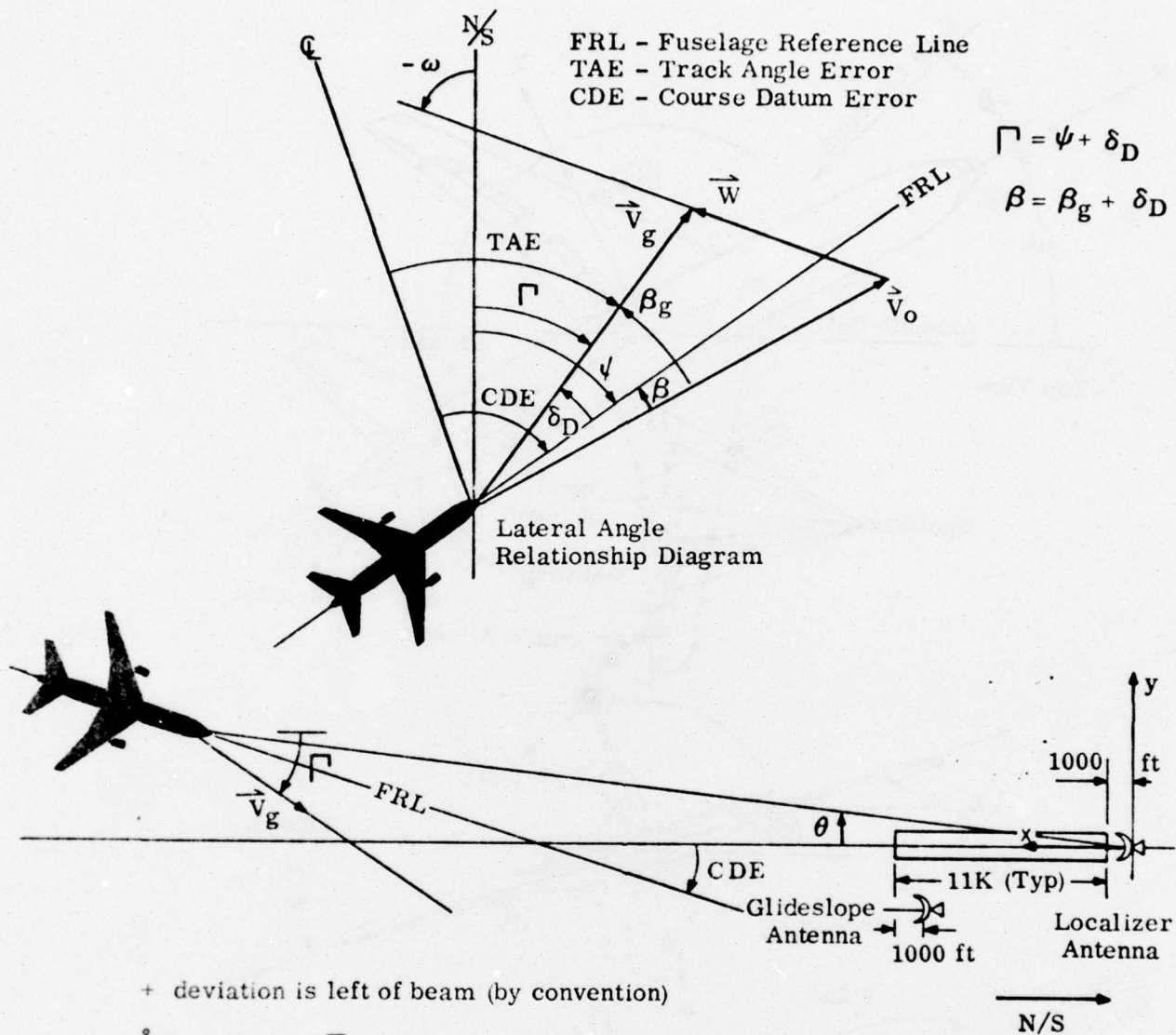
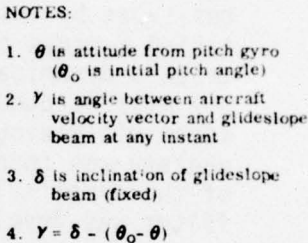


Figure 38. ILS Localizer Geometry.



97

in the capture computations. The capture computations can be visualized as computing a circle which would be tangential to the aircraft flight path and to the center line of the radio beam to be acquired. By utilizing a predictive-corrective technique, it is feasible to compute the aircraft ground velocity and the mean wind allowing for compensation of these terms during the capture. The key to the computations is to derive good estimates of beam position and beam rate. These terms are central to the estimation of distance from the station of aircraft ground velocity and of wind.

The key challenge in the estimation process was the necessity to obtain derivative information from noisy beam data. The derivative information is required for two reasons: 1) the only available data relating to range-to-station is contained in the ratio of beam and beam rate at the given aircraft ground velocity, and 2) rate damping is needed in the track laws. To avoid enhancing the high frequency noise present in the signal, the derivative-taking process must be restricted to low frequencies. However, the derivative-taking process must extend to high enough frequencies to adequately track a dynamically changing beam. Typically, a performance trade-off must be considered between tracking error and noise error since minimizing one enhances the other. In the analysis, an advantage was found to exist in using a filter which contains a model of the dynamical process such as a Kalman filter. Hence, a Kalman filter was developed to perform the estimation process of deriving beam and beam rate information. Once a Kalman filter is essentially tracking the process, it has a better capability than a simpler filter to reject noise without excessive tracking error. However, these filters must be initialized (i.e., the dynamical model must have a starting point). In the initialization process they are also subject to noise. If quick initialization is attempted, noise errors grow; while if a slow initialization is permitted, tracking errors grow. Clearly for the capture maneuver initialization time is limited. Initialization cannot start until the beam is intercepted, a beam deviation* greater than $2.75-3.0^\circ$ does not yield usable radio information, and it must be complete before the radio deviation becomes too small. The signal to noise ratio of the radio deviation decreases as the beam angle decreases.

Almost any control analysis is confronted with two distinct considerations, the physical control of the plant, in this case an aircraft, and the measurement or estimation of the state parameters needed to implement the control laws. Hence Figure 40 has sought to initially separate the control analysis from the estimation analysis. In this way each may be understood in its own right without the confusion of cross-coupling effects. In the following description and sections, the control and estimation processes are discussed separately but the interface between them is completely described.

Figure 40 provides a conceptual view of the total lateral axis control-estimation process for the DFDC (the longitudinal axis is directly analogous). Figure 40 illustrates in a block diagram form how the

* For TACAN, localizer, and glideslope, the beam limits are approximately 10° , 3° , and 1° , respectively.

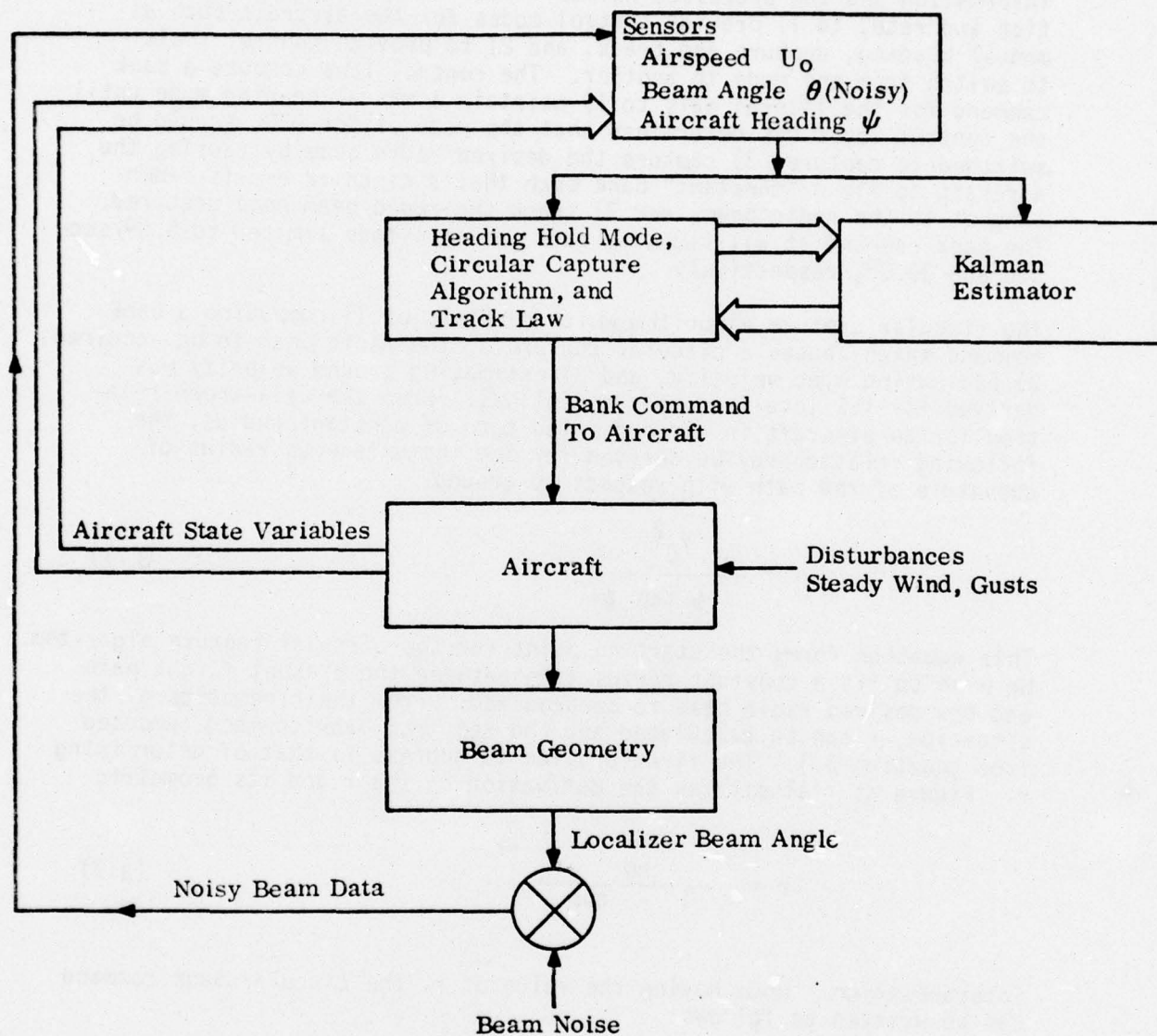


Figure 40. Total Lateral Axis Control-Estimation Process for the Digital Flight Director Computer.

major elements of the control-estimation process are interrelated to one another. First, the sensors provide information as to the state of the aircraft relative to a radio beam that is to be acquired. The Kalman filter uses the sensor information and control law state to provide a best estimate of position and position rate information to the control laws. The control laws use the unprocessed sensor information and the processed Kalman filter sensor information, position and rate, to 1) provide control modes for the aircraft such as manual heading, capture and track, and 2) to provide control logic to switch from one mode to another. The control laws compute a bank command for the lateral axis to 1) maintain a manual heading mode until the control logic has determined that the mode of the DFDC should be switched to capture, 2) capture the desired radio beam by causing the aircraft to fly a "constant" bank such that a circular arc is flown tangent to the radio beam, and 3) track the radio beam once acquired. The bank command at all times is rate and amplitude limited to 5.0°/sec and 30.0°, respectively.

The circular capture algorithm which consists of 1) computing a bank command which causes a circular capture of the radio beam to be acquired, 2) estimating wind velocity, and 3) estimating ground velocity was derived for ILS localizer mode as follows. From the well-known relation for an aircraft in a coordinated turn of constant radius, the following relation may be derived for the instantaneous radius of curvature of the path with respect to ground.

$$r = \frac{V_g^2}{g \tan \phi} \quad (3.1)$$

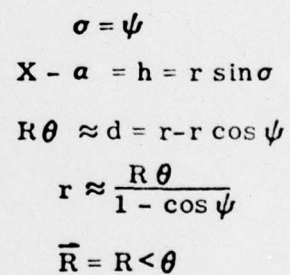
This equation forms the starting point for the circular capture algorithm. We wish to fit a constant radius turn between the present flight path and the desired radio beam to be acquired. From the kinematics of the situation, r can be calculated and the required bank command computed from equation 3.1. The first problem to address is that of determining r . Figure 41 illustrates the derivation of the r and its geometric

$$r = \frac{R\theta}{1 - \cos \theta} \quad (3.2)$$

interpretation. Once having the value of r , the circular bank command can be written as follows:

$$\phi = \tan^{-1} \frac{V_g^2}{rg} \quad (3.3)$$

If the quantities in these equations were available error free, they could be applied directly. However, several problems exist. Ground speed, V_g , is not readily available, although airspeed $(U_0)_k$ is. There will be errors in measuring ψ , course datum error. Range, r , is not readily available in any direct measurement except for a ratio of beam and beam rate. Finally, in taking the derivative of beam deviation θ to get $\dot{\theta}$, errors will result since θ is typically noisy. Another



101

factor is that turbulence, steady winds and wind shear would cause an open loop control calculation at a single point to perform poorly. To overcome the latter problem, a multistage decision process was employed where new data is taken and a new command computed every J seconds. The multistage decision process is illustrated in Figure 42. (Figure 43 illustrates the computation and geometric interpretation of $\vec{e}_k, \hat{W}_k, \hat{\psi}_k$.) The result of the first step of the process is the computation of the apparent position error vector, \vec{e}_k . Step two uses the apparent position error vector to generate an estimate of the wind vector, \hat{W}_k . The vector equation for \hat{W}_k is:

$$\hat{W}_k = \hat{W}_{k-1} + \frac{\lambda \vec{e}_k}{J} \quad (3.4)$$

Where \vec{e}_k is the apparent position error vector, J is the sample time, and λ is a proportionality constant whose optimum value was determined from hybrid simulation. Thus, the best estimate of ground velocity becomes:

$$(\hat{V}_g)_k = (\hat{U}_0)_k + \hat{W}_k \quad (3.5)$$

Finally, step three results in the computation of the bank command which results in the optimal circular capture. Figure 44 illustrates the flow chart and equations for the originally proposed ILS localizer mode circular capture algorithm. These equations and flow chart illustrate the measurement, prediction, and estimation process required to derive the best estimate of wind, ground velocity, and the computation of the circular bank command, $\hat{\phi}_k$.

Figure 45 illustrates the current circular capture algorithm for the ILS localizer mode. The flow chart and equations define the computational process for estimating wind, ground velocity, and flight path angle resulting in the circular capture bank command, $\hat{\phi}_k$. The differences in Figures 44 and 45 are discussed in the following.

The computation of range-to-station, equations 3.8-1,* and 3.8-3, was modified to equation 3.9-2. The implementation of equation 3.9-2 results in a measurement of range-to-station, R_k , by using the best estimates of ground velocity, flight path angle, beam, and beam rate. The measured range is then smoothed through a predictive-corrective type filter.

The computational method of equation 3.8-2 was modified by substituting equation 3.1 resulting in equation 3.9-1. This equation provides feedback in the form of the measured bank of the aircraft as to the

* The notation 3.8-1 refers to Figure 44, equation 1.

STEP ONE

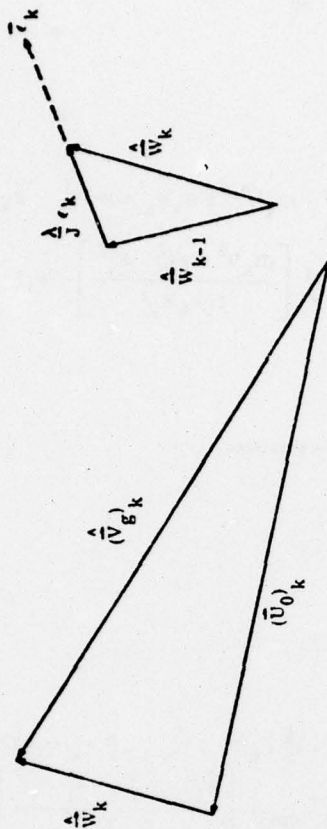
1. Predict R_k^1, θ_k^1
2. Measure $\theta_k, \psi_k, (\hat{U}_0)_k$
3. Compute R_k
4. Compute $\hat{\tau}_k = \hat{\tau}_k(R_k, \theta_k, R_k^1, \theta_k^1)$

STEP TWO

$$\text{Estimate } (\hat{V}_g)_k = (\hat{U}_0)_k + \hat{W}_k$$

where $\hat{W}_k = \hat{W}_{k-1} + \frac{1}{j} \hat{\tau}_k$ is the computed

"wind" vector (see note 1) -- $\hat{W}_1 \triangleq \vec{0}$



STEP THREE

1. Compute estimate of distance from transmitter (\hat{R}_k)
2. Compute Control $\hat{\phi}_k = \hat{\phi}_k(\hat{R}_k, \theta_k, (\hat{V}_g)_k, \hat{\psi}_k)$

NOTES:

1. \hat{W}_k also compensates for measurement errors
2. Primes indicate predicted variables (e.g. R_k^1)
3. "Hats" indicate "best" estimates of variables (e.g. \hat{R}_{k-1})

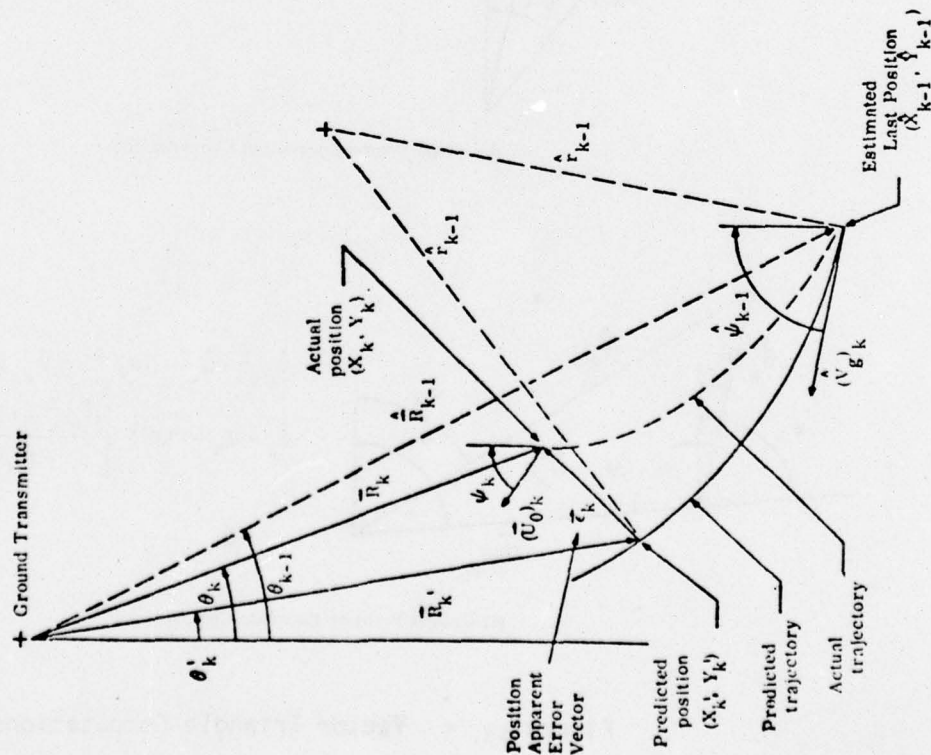
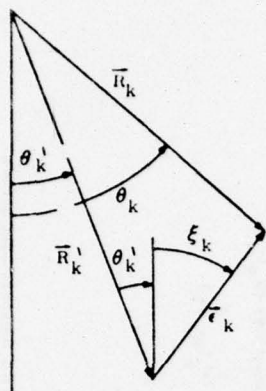


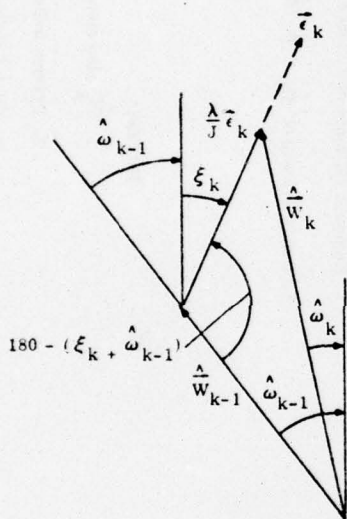
Figure 42. Overview of Multistage Decision Process.



(a) Error Vector Computations

$$\epsilon_k^2 = R_k^2 + (R_k')^2 - 2 R_k R_k' \cos (\theta_k - \theta_k')$$

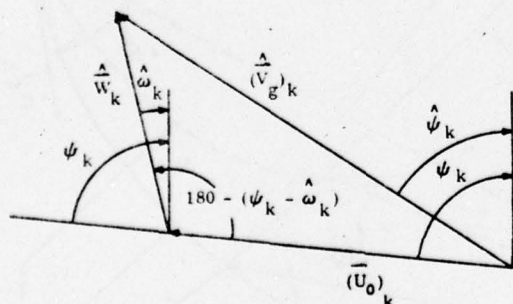
$$\xi_k = \cos^{-1} \left[\frac{(R_k')^2 + \epsilon_k^2 - R_k^2}{2 \epsilon_k R_k'} \right] - \theta_k'$$



(b) Wind Vector Correction Computations

$$\hat{W}_k^2 = \hat{W}_{k-1}^2 + \left(\frac{\lambda}{J} \epsilon_k\right)^2 + 2 \hat{W}_{k-1} \left(\frac{\lambda}{J} \epsilon_k\right) \cos (\xi_k - \hat{\omega}_{k-1})$$

$$\hat{\omega}_k = \hat{\omega}_{k-1} - \cos^{-1} \left[\frac{\hat{W}_k^2 + \hat{W}_{k-1}^2 - \left(\frac{\lambda}{J} \epsilon_k\right)^2}{2 \hat{W}_k \hat{W}_{k-1}} \right]$$

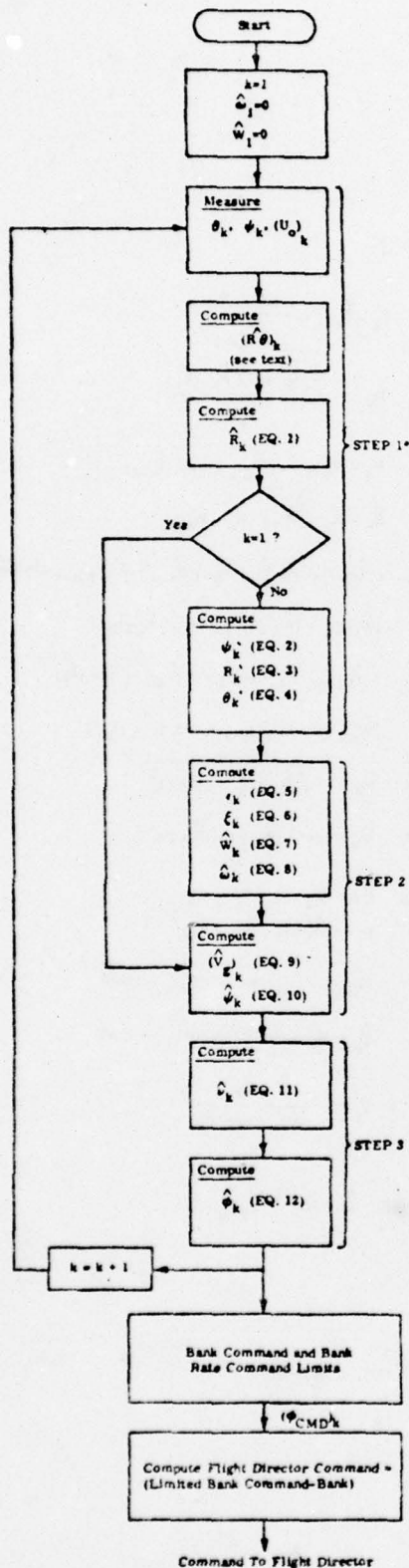


(c) Ground Velocity Estimate Computations

$$(\hat{V}_g)_k^2 = \hat{W}_k^2 + (U_0)_k^2 + 2 \hat{W}_k (U_0)_k \cos (\psi_k - \hat{\omega}_k)$$

$$\hat{\psi}_k = \psi_k - \cos^{-1} \left[\frac{(\hat{V}_g)_k^2 + (U_0)_k^2 - \hat{W}_k^2}{2 (\hat{V}_g)_k (U_0)_k} \right]$$

Figure 43. Vector Triangle Computations.



$$1. \hat{R}_k = \frac{(R\theta)_k}{\theta_k}$$

$$2. \psi_k = \hat{\psi}_{k-1} + \frac{(\hat{V}_g)_{k-1}}{\hat{r}_{k-1}} J$$

$$3. R_k = \hat{R}_{k-1} - \hat{r}_{k-1} (\sin \hat{\psi}_{k-1} - \sin \psi_k)$$

$$4. \theta_k = \frac{\hat{\theta}_{k-1}}{R_k} (1 \pm \cos \psi_k)$$

$$5. \epsilon_k = \sqrt{R_k^2 + (R_k')^2 - 2 R_k R_k' \cos (\theta_k - \theta_k')}$$

$$6. \xi_k = \cos^{-1} \left(\frac{(R_k')^2 + \epsilon_k^2 - R_k^2}{2 \epsilon_k R_k'} \right) - \theta_k'$$

$$7. \hat{W}_k = \sqrt{\left(\frac{\Delta \epsilon_k}{J} \right)^2 + \hat{W}_{k-1}^2 + 2 \hat{W}_{k-1} \left(\frac{\Delta \epsilon_k}{J} \right) \cos (\xi_k + \hat{\psi}_{k-1})}$$

$$8. \hat{\psi}_k = \hat{\psi}_{k-1} - \cos^{-1} \left(\frac{\hat{W}_k^2 + \hat{W}_{k-1}^2 - \left(\frac{\Delta \epsilon_k}{J} \right)^2}{2 \hat{W}_k \hat{W}_{k-1}} \right)$$

$$9. (\hat{V}_g)_k = \sqrt{\hat{W}_k^2 + (U_o)_k^2 + 2 \hat{W}_k (U_o)_k \cos (\psi_k - \hat{\psi}_k)}$$

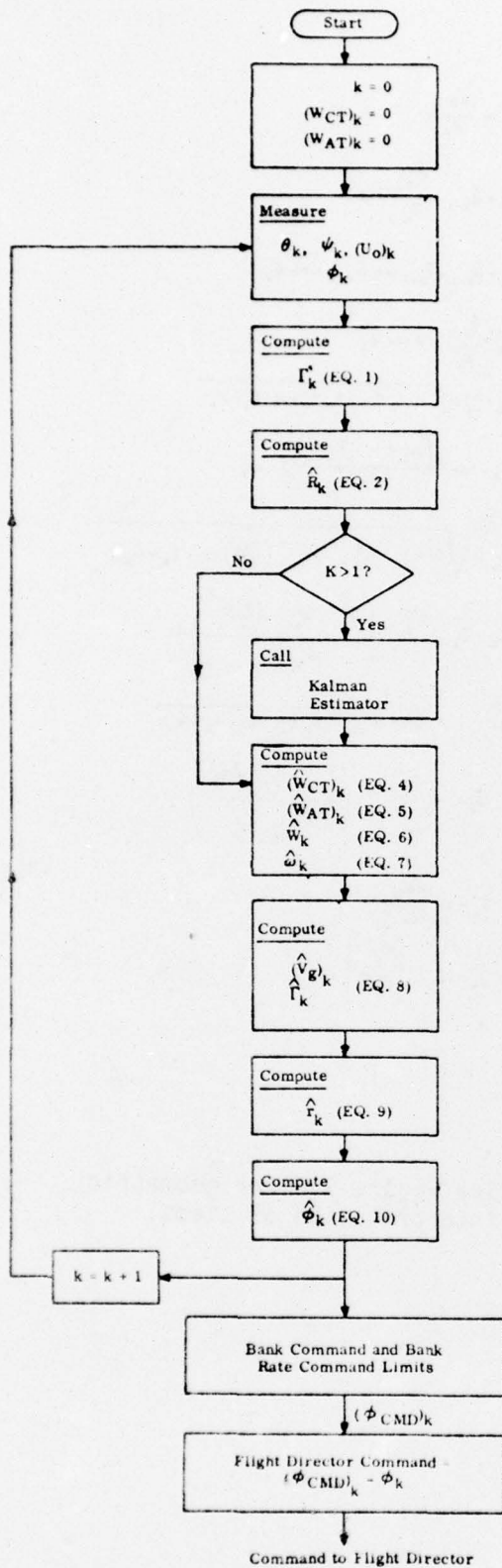
$$10. \psi_k = \psi_k - \cos^{-1} \left(\frac{(\hat{V}_g)_k^2 + (U_o)_k^2 - \hat{W}_k^2}{2 (\hat{V}_g)_k (U_o)_k} \right)$$

$$11. \hat{r}_k = \frac{(R\theta)_k}{1 \pm \cos \hat{\psi}_k}$$

$$12. \hat{\phi}_k = \tan^{-1} \left[\frac{(\hat{V}_g)_k^2}{\hat{r}_k \xi_k} \right]$$

*See Figure 3.6 for geometric interpretation of steps.

Figure 44. Originally Proposed ILS (Localizer) Mode Circular Capture Algorithm.



1. $\hat{\Gamma}_k^s = \hat{\Gamma}_{k-1}^s + \frac{J g \tan \phi_{k-1}}{(\hat{V}_g)_{k-1}}$
2. $R_k^s = \frac{(\hat{V}_g)_k \sin(\hat{\theta}_k - \hat{\Gamma}_k^s)}{\hat{\theta}_k}$
 $R_k^s = \hat{R}_{k-1}^s - J (\hat{V}_g)_{k-1} \cos(\hat{\Gamma}_{k-1}^s)$
 $\hat{R}_k^s = R_k^s + \lambda_{AY} (R_k - R_k^s)$
4. $(\hat{\epsilon}_{CT})_k = \hat{R}_k \hat{\theta}_k - \hat{R}_{k-1} \hat{\theta}_{k-1} + J (\hat{V}_g)_{k-1} \sin(\hat{\Gamma}_k^s)$
 $(\hat{W}_{CT})_k = (\hat{W}_{CT})_{k-1} + \lambda_c (\hat{\epsilon}_{CT})_k / J$
5. $(\hat{\epsilon}_{AT})_k = \hat{R}_k - \hat{R}_{k-1} + J (\hat{V}_g)_{k-1} \cos(\hat{\Gamma}_k^s)$
 $(\hat{W}_{AT})_k = (\hat{W}_{AT})_{k-1} + \lambda_A (\hat{\epsilon}_{AT})_k / J$
6. $\hat{W}_k = \sqrt{(\hat{W}_{AT})_k^2 + (\hat{W}_{CT})_k^2}$
7. $\hat{\omega}_k = \tan^{-1} (\hat{W}_{CT})_k / (\hat{W}_{AT})_k$
8. $CW = \hat{W}_k \cos(\hat{\theta}_k - \psi_k)$
 $SW = \hat{W}_k \sin(\hat{\theta}_k - \psi_k)$
 $(\hat{V}_g)_k = \sqrt{(SW)^2 + ((U_o)_k + CW)^2}$
 $\hat{\Gamma}_k^s = \psi_k + \tan^{-1} (SW / ((U_o)_k + CW))$
9. $\hat{\Gamma}_k = \frac{\hat{R}_k \hat{\theta}_k}{1.0001 - \cos \hat{\Gamma}_k}$
10. $\hat{\phi}_k = \tan^{-1} \left[\frac{(\hat{V}_g)_k^2}{\hat{R}_k g} \right]$

(EQ. 3. deleted)

Figure 45. Current ILS (Localizer) Mode Circular Capture Algorithm.

actual circular arc that is being flown. This feedback provides correcting information during that time when the pilot desires not to implement the bank steering command.

The wind computation consists of equation 3.8-4 through 3.8-8 and has been converted from a magnitude, angular computation to equations 3.9-4 through 3.9-7, consisting of cross track, and of along track wind computations. The reason for the modification was a result of 1) computing the difference of too "large" numbers in equation 3.8-5, and 2) requiring the implementation of the inverse cosine function.

The control-estimation equations for the longitudinal axis were developed in a very similar manner to those of the lateral axis. Referring to Figure 39 and noting the following similarities: $\delta \sim \psi$; $\zeta \sim \theta$; $S \sim R$, $V_T \sim V_g$, $r \sim \eta$, the equations for the longitudinal case can immediately be written except for the command equation. Recalling that the constant bank command implies constant heading rate, the form of the command is indicated by

$$\dot{\Omega} = \frac{V_T}{\eta} \quad (3.6)$$

where Ω is the pitch angle. That is, one should command constant pitch rate to fly a circular segment in the vertical plane. The resulting command equations are:

$$r = \frac{(S_\zeta)}{1 - \cos \delta} \quad (3.7)$$

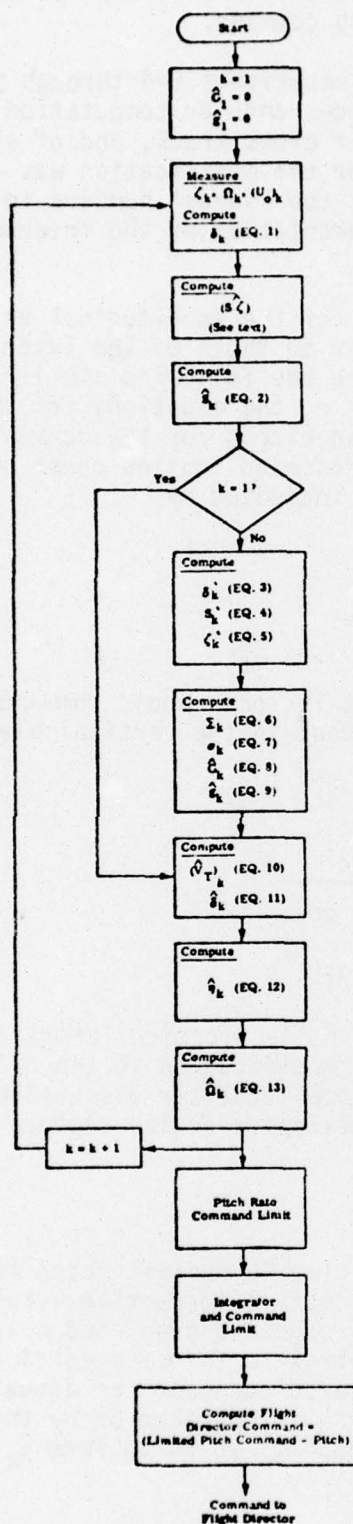
$$\dot{\Omega} = V_T/r$$

Figure 46 illustrates the originally proposed ILS glideslope circular capture algorithm. The modification to the glideslope algorithm follows directly from the localizer discussion of Paragraph 3.1. The current ILS glideslope circular capture algorithm is illustrated in Figure 47.

3.2 Kalman Estimation Process

A block diagram of the discrete time Kalman estimator is illustrated in Figure 48. Notice, the predictive-corrective nature of the estimator, the dynamical model produces a predicted next state \hat{x}_k which is multiplied by the H_k matrix to form a predicted measurement Z_k . The difference between the predicted and the actual measurements, "the apparent state error vector", is operated on by the Kalman gains, K_k , and is used to "correct" \hat{x}_k so as to form \hat{x}_k , the best estimate state vector.

The detailed matrix difference equations for the dynamical model in the Kalman estimator are illustrated in Figure 49. Notice, the definition of the state vector, x . x_1 is (R_θ) , cross track distance.



$$1. \delta_k = \gamma - (D_o - D_k)$$

$$2. \hat{\delta}_k = \frac{\delta \zeta_k}{\zeta_k}$$

$$3. \delta_k' = \hat{\delta}_{k-1} + \frac{(\hat{V}_T)_{k-1}}{\delta_{k-1}} \delta$$

$$4. \delta_k' = \hat{\delta}_{k-1} - \hat{\delta}_{k-1} (\sin \hat{\delta}_{k-1} - \sin \delta_k')$$

$$5. \zeta_k' = \frac{\hat{\zeta}_{k-1}}{\delta_k'} (1 + \cos \delta_k')$$

$$6. \Sigma_k = \sqrt{\delta_k'^2 + (\zeta_k')^2} - 2 \delta_k' \zeta_k' \cos (\zeta_k - \zeta_k')$$

$$7. \sigma_k = \cos^{-1} \left(\frac{(\delta_k')^2 + \Sigma_k^2 - \delta_k^2}{2 \delta_k' \Sigma_k} \right) - \zeta_k'$$

$$8. \hat{C}_k = \sqrt{\left(\frac{\delta \Sigma_k}{\delta} \right)^2 + \hat{C}_{k-1}^2} + 2 \hat{C}_{k-1} \left(\frac{\delta \Sigma_k}{\delta} \right) \cos (\sigma_k + \hat{\delta}_{k-1})$$

$$9. \hat{\delta}_k = \hat{\delta}_{k-1} - \cos^{-1} \left(\frac{\hat{C}_k^2 - \hat{C}_{k-1}^2 - \left(\frac{\delta \Sigma_k}{\delta} \right)^2}{2 \hat{C}_k \hat{C}_{k-1}} \right)$$

$$10. (\hat{V}_T)_k = \sqrt{\hat{C}_k^2 + (U_o)_k^2} + 2 \hat{C}_k (U_o)_k \cos (\delta_k - \hat{\delta}_k)$$

$$11. \hat{\delta}_k = \delta_k - \cos^{-1} \frac{(\hat{V}_T)_k^2 + (U_o)_k^2 - \hat{C}_k^2}{2(\hat{V}_T)_k (U_o)_k}$$

$$12. \hat{\delta}_k = \frac{\delta \zeta_k}{1 + \cos \hat{\delta}_k}$$

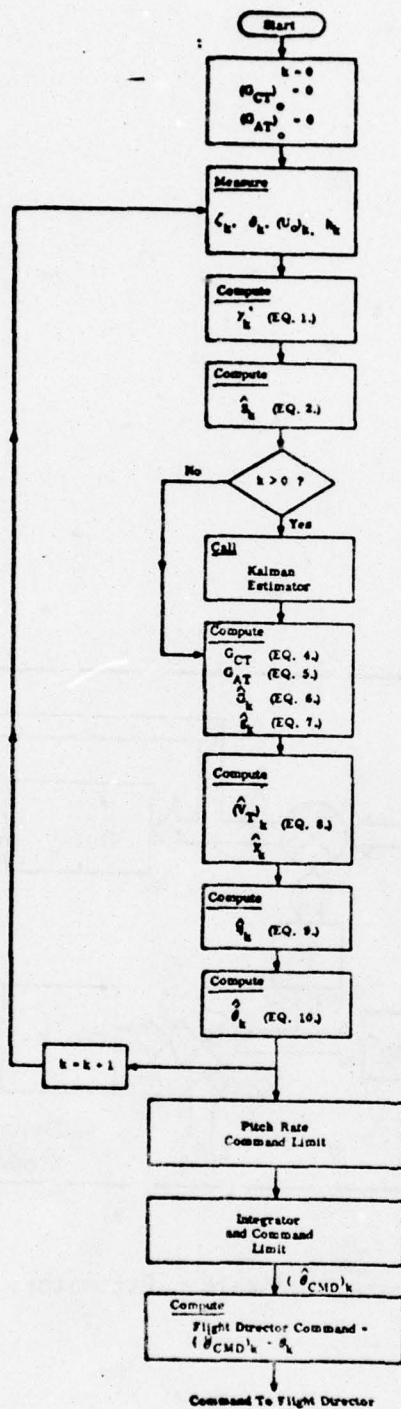
$$13. \hat{\delta}_k = \frac{(\hat{V}_T)_k}{\delta_k}$$

Notes: $\Sigma_k = \Sigma_k \angle \sigma_k$ Position Error Vector

$\hat{C}_k = \hat{C}_k \angle \hat{\delta}_k$ Wind Vector

$(\hat{V}_T)_k = \hat{V}_T \angle \delta$ Aircraft Velocity Vector (beam coordinates)

Figure 46. Originally Proposed ILS (Glideslope) Mode Circular Capture Algorithm.



1. $\hat{Y}_k = \hat{Y}_{k-1} + J \hat{d}_k$
2. $\hat{d}_k = \hat{d}_{k-1} / \tan^{-1}(\delta - \hat{z}_k)$
 $\hat{d}_k = \hat{d}_{k-1} - J(\hat{V}_T)_k \cos(\hat{Y}_k)$
 $\hat{d}_k = \hat{d}_k + 0.1(\hat{d}_{k-1} - \hat{d}_k)$
3. $(CT)_k = \hat{z}_k - \hat{z}_{k-1} - \hat{z}_{k-1} + J(\hat{V}_T)_{k-1} \sin(\hat{Y}_k)$
 $(CT)_k = (CT)_{k-1} + \lambda_{CT} (CT)_k / J$
4. $(AT)_k = \hat{z}_k - \hat{z}_{k-1} - J(\hat{V}_T)_{k-1} \cos(\hat{Y}_k)$
 $(AT)_k = (AT)_{k-1} + \lambda_{AT} (AT)_k / J$
5. $\hat{Q}_k = \sqrt{(CT)_k^2 + (AT)_k^2}$
6. $\hat{Q}_k = \tan^{-1}((CT)_k / (AT)_k)$
7. $CW = \hat{Q}_k \cos(\hat{Q}_k - \delta - \theta_k)$
 $SW = \hat{Q}_k \sin(\hat{Q}_k - \delta - \theta_k)$
 $(\hat{V}_T)_k = \sqrt{SW^2 + ((U_d)_k - CW)^2}$
 $\hat{Y}_k = \theta_k + \delta + \tan^{-1}\left(\frac{SW}{(U_d)_k - CW}\right)$
8. $Q_k = \frac{\hat{Q}_k \hat{Q}_k}{1.0001 - \cos(\hat{Y}_k)}$
9. $\hat{Q}_k = \frac{(\hat{V}_T)_k}{Q_k}$

(EQ. 3 deleted)

Figure 47. Current ILS (Glideslope) Mode Circular Capture Algorithm.

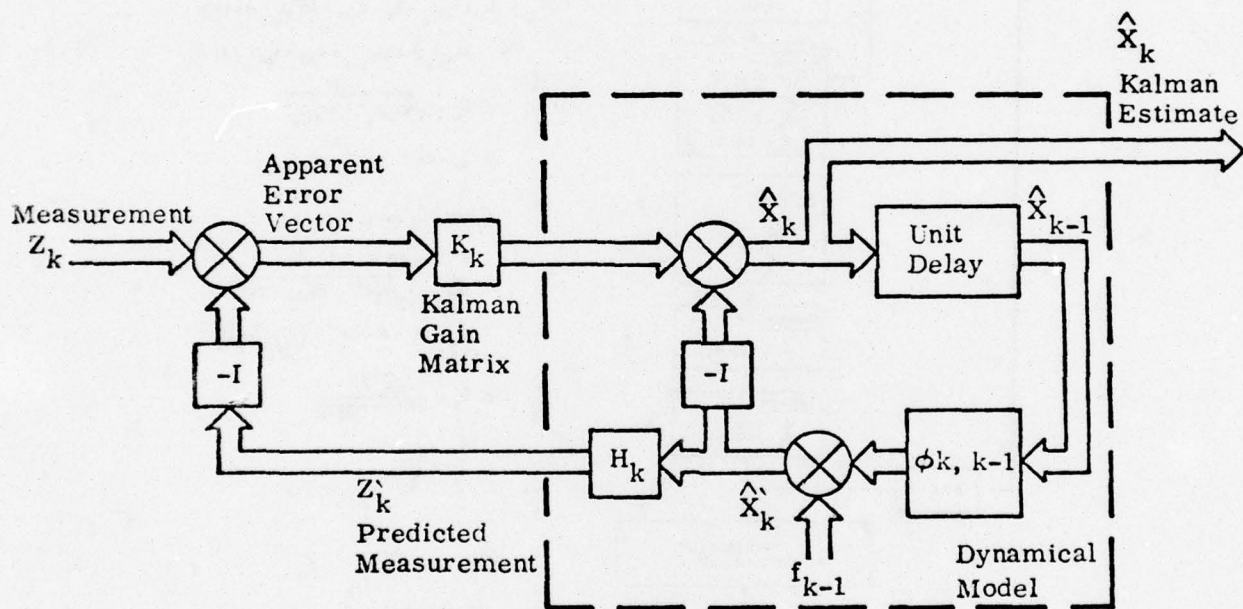


Figure 48. Discrete Time Kalman Estimator.

Dynamical Model

$$\begin{pmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \end{pmatrix} = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & e^{-T/4} & 0 \\ 0 & 0 & 0 & e^{-T/2} \end{pmatrix} \begin{pmatrix} X_1(k-1) \\ X_2(k-1) \\ X_3(k-1) \\ X_4(k-1) \end{pmatrix} + \begin{pmatrix} 0 \\ -gT \tan \phi \cos \Gamma \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \rho_2(k) \\ \rho_3(k) \\ \rho_4(k) \end{pmatrix}$$

X_1 = Cross Track Distance

X_2 = Cross Track Rate

X_3 = Beam Noise (angular)

X_4 = Gust Noise (linear)

Measurement Model

$$Z(k) = (1, 0, \hat{R}_k, 0) \begin{pmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \end{pmatrix} + v(k)$$

Covariance Matrix

$$Q_k = E \left[\begin{pmatrix} 0 \\ \rho_2(k) \\ \rho_3(k) \\ \rho_4(k) \end{pmatrix} \begin{pmatrix} 0, \rho_2(k), \rho_3(k), \rho_4(k) \end{pmatrix} \right] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & E[\rho_2^2] & 0 & 0 \\ 0 & 0 & E[\rho_3^2] & 0 \\ 0 & 0 & 0 & E[\rho_4^2] \end{bmatrix}$$

$$R_k = E[v^2(k)]$$

Figure 49. Position, Position Rate (y, \dot{y}) Kalman Estimator.

x_2 is $(R'\theta)$, cross track rate. The radio beam noise is modeled by x_3 which has a four second correlation characteristic. Gust activity moves the aircraft in a noise-life manner. The resulting aircraft acceleration is modeled by x_4 which is a two second correlated noise sequence.

A feature that is included in the measurement model is a measurement noise v_k (assumed white) with covariance matrix R_k . R_k is a 1×1 matrix or scalar. Intuitively, the measurement noise matrix provides the filter with some information relative to the signal to noise ratio of the incoming data. This is clearer if one reviews the way the Kalman gains are generated. The optimal state estimate at time t_k will be given by the state estimation equations as

$$\hat{x}_k = \hat{x}_k^- + K_k (Z_k - H_k \hat{x}_k^-) \quad (3.8)$$

$$\hat{x}_k^- = \phi_{k,k-1} \hat{x}_{k-1} + f_{k-1}$$

Where (referring to Figure 49) $\phi_{k,k-1}$ is the transition matrix. f_{k-1} is the forcing term in the $\phi_{k,k-1}$ dynamical model (discussed above) and K_k is the Kalman gain matrix. Notice that when elements of K_k are large, much confidence is being placed in the measured data and conversely when they are small, less weight is being given to the received data. To see that the measurement noise covariance matrix accomplishes this desired end, examine the recursion relations that generate the Kalman gain matrix, equation 3.9a.

$$\begin{aligned} P_k^- &= \phi_{k,k-1} P_{k-1} \phi_{k,k-1}^T + Q_{k-1} \\ K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \end{aligned} \quad (3.9a)$$

$$P_k = P_k^- - K_k H_k P_k^-$$

$$\begin{aligned} E \left[\rho_k \rho_j^T \right] &= Q_k \\ E \left[x_0 \right] &= \bar{x}_0 \\ E \left[(x_0 - \bar{x}_0) (x_0 - \bar{x}_0)^T \right] &= P_0 \\ E \left[\rho_k x_0^T \right] &= 0 \text{ for all } k \end{aligned} \quad (3.9b)$$

Since R_k occurs in a matrix summation and that summation is subsequently inverted and used as one of the terms of a matrix product to generate the gain matrix K_k , then making the elements of K_k large will tend to decrease the gain and the converse is also true. It has been found that making some elements of the Q_k matrix large will tend to keep the Kalman gains initially high, aiding in the initial convergence and tracking during the capture maneuver. As track mode approaches, increasing R_k , decreases the gains reflecting 1) the lower signal to noise ratio in track and 2) the lower dynamic bandwidth required in track. These factors are not contradictory; they allow lower gains in track, permitting the bank activity to be held to a minimum. Equation 3.9b provides the initial conditions to the recursive equation 4.9a. The Kalman filter solution is the one that minimizes

$$E \left[(\hat{x}_k - x_k)^T, (\hat{x}_k - x_k) \right].$$

The white noise sequences that occur in the dynamical model for the Kalman estimator are illustrated in Figure 49. Basically, it is the specified noise covariances, dynamical and measurement models that determine the behavior of the filter, that is how the Kalman gains are caused to vary. The model was determined by physical reasoning and the covariances were determined from the noise models. In this model, the variances of the discrete white noise sequences were set to produce the proper variance of the colored noise sequences. In the case of X_3 , this implies that $E \left[\rho_3^2 \right] = Q_{33}$ must be set so that $E \left[(X(k))^2 \right]$ is equal to the variance of the beam noise assuming a zero mean process. The general relationship

$$Q_{ii} = \sigma_i^2 (1 - e^{-2T/\tau}) \quad (3.10)$$

was used to determine the covariance matrix where σ_i^2 is the variance of the continuous correlated noise. T is the sample time of the discrete model, and τ is the correlation time of the noise.

3.2.1 TACAN Kalman Estimation Process

The Kalman estimator for TACAN retains the originally proposed position, position rate form $(R\theta, R\dot{\theta})$ as shown in Figure 49, since DME range information is available to supply range data. The specific Q_k and R_k matrices have been tailored to this particular form. The state variables are as defined in Figure 49.

3.2.2 ILS Kalman Estimation Process

The basic Kalman estimation algorithm was modified from a cross track distance, cross track rate form, $(R\theta, R\dot{\theta})$ as described in Phase I final report to an angular position, angular rate form $(\theta, \dot{\theta})$ for ILS modes. Figure 50 illustrates the total ILS lateral axis control-estimation process. Several factors have influenced the change from the $(R\theta, R\dot{\theta})$ form to the $(\theta, \dot{\theta})$ form. The original Kalman estimator would have been a natural choice if ILS range data would have been available. Since ILS range data was not available, a pre-filter

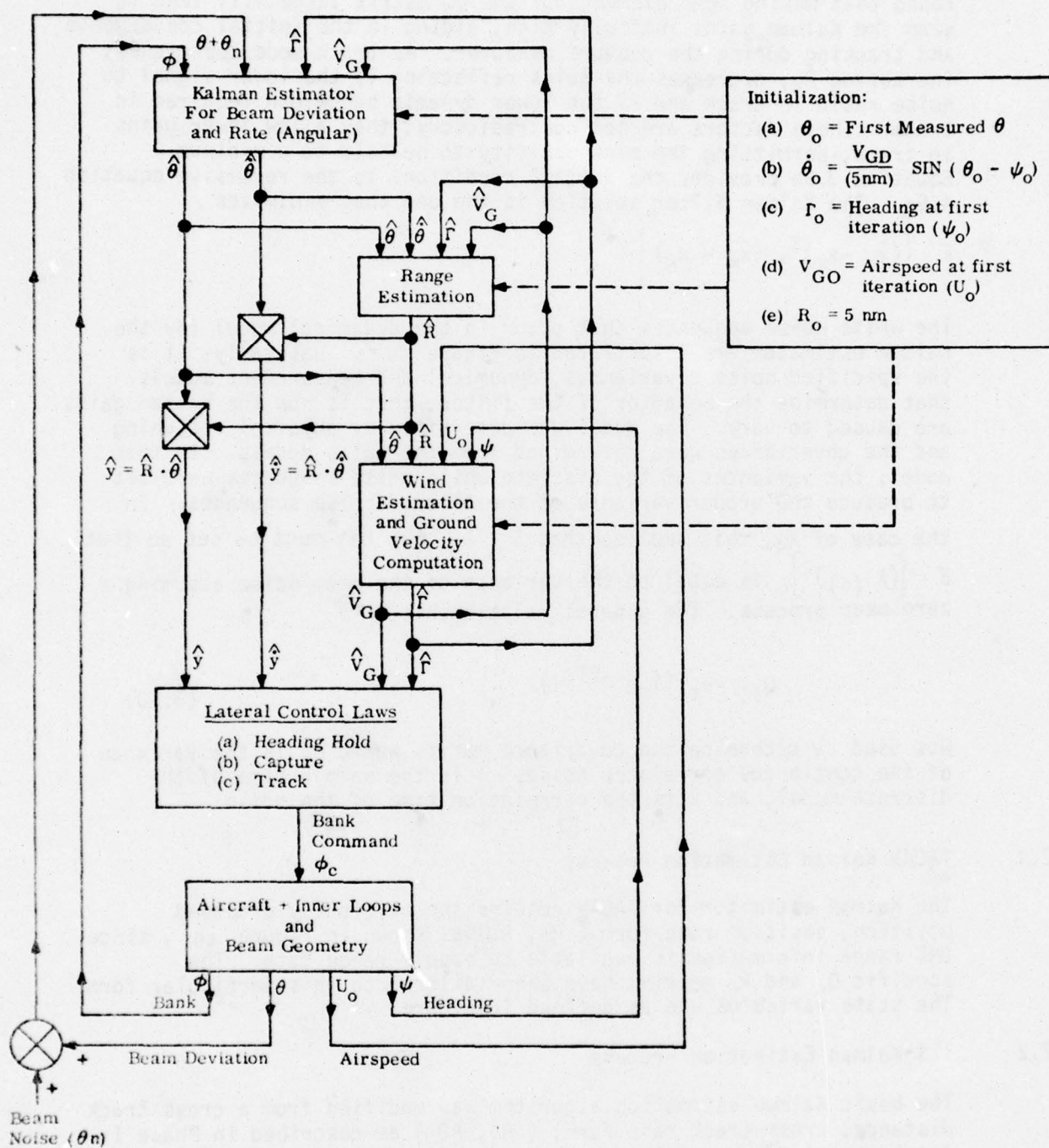


Figure 50. Block Diagram of the Total ILS Lateral Axis Control-Estimation Process.

was required to derive range to station information and to provide initialization of the Kalman estimator. Analysis and simulation results revealed that the α - β tracker pre-filter did not provide adequate simultaneous suppression of beam noise and tracking error during the dynamics of the capture maneuver. Close tracking of beam rate was found to be essential if wind and range errors were to be separated during the capture maneuver. Other pre-filters were also extensively studied and it was concluded that the algorithm of Figure 50 which does not require a pre-filter, performs satisfactorily and was thus selected. This algorithm has proven that it maintains comparable performance to that of the originally proposed Kalman estimator in the ILS modes. The state variables for the ILS modes are defined in Figure 51. X_1 is θ , beam position. X_2 is $\dot{\theta}$, beam rate. X_3 is the beam noise. X_4 is the gust activity.

3.2.2.1 Localizer Kalman Estimation Process

Figure 51 illustrates the angular, angular rate form of the ILS mode (localizer) Kalman estimator. The primary difference between this form and the position, position rate estimator used for TACAN is the form of the forcing term in the $X_2(\kappa)$ equation. To see how this arises, let

$$\ddot{y} = g \tan(\phi) \cos(\Gamma) \quad (3.11)$$

where \ddot{y} is the acceleration of the aircraft relative to the localizer beam, ϕ and Γ are bank and track angle respectively. If the approximation

$$\ddot{y} \approx R\ddot{\theta} \quad (3.12)$$

is used where $\ddot{\theta}$ is angular acceleration relative to the beam, then

$$\ddot{\theta} = g \tan(\phi) \cos(\Gamma)/R \quad (3.13)$$

this equation may be discretized as

$$\theta(\kappa) = \theta(\kappa-1) + J g \tan(\phi) \cos(\Gamma)/R \quad (3.14)$$

equating $X_2(\kappa) = \theta(\kappa)$ and adding the gust noise contribution yields the formulation illustrated in Figure 51. The validity of the above approximations has been demonstrated in hybrid simulation results, which will be discussed in later sections of this report. In addition, elimination of the pre-filter permits a simpler and more direct synthesis of the Kalman estimator. The discussion of paragraph 3.2 relative to the use of Q_κ and R_κ matrices to tailor the Kalman gains also applies.

3.2.2.2 Glideslope Kalman Estimation Process

The form of the glideslope Kalman estimator is identical to the localizer form shown in Figure 51, except for the forcing term in the $X_2(\kappa)$ equation. The forcing term for the longitudinal axis was

Dynamical Model

$$\begin{pmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \end{pmatrix} = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & e^{-T/4} & 0 \\ 0 & 0 & 0 & e^{-T/2} \end{pmatrix} \begin{pmatrix} X_1(k-1) \\ X_2(k-1) \\ X_3(k-1) \\ X_4(k-1) \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{-gT \tan \phi \cos \Gamma}{R} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \eta_2(k) \\ \eta_3(k) \\ \eta_4(k) \end{pmatrix}$$

X_1 = Angular Beam Deviation

X_2 = Angular Rate

X_3 = Beam Noise

X_4 = Gust Noise

Measurement Model

$$Z(k) = (1, 0, 1, 0) \begin{pmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \end{pmatrix} + v(k)$$

Covariance Matrix

$$Q_k = E \left[\begin{pmatrix} 0 \\ \eta_2(k) \\ \eta_3(k) \\ \eta_4(k) \end{pmatrix} \begin{pmatrix} 0, \eta_2(k), \eta_3(k), \eta_4(k) \end{pmatrix} \right] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & E(\eta_2^2) & 0 & 0 \\ 0 & 0 & E[\eta_3^2] & 0 \\ 0 & 0 & 0 & E[\eta_4^2] \end{bmatrix}$$

$$R_k = E[v^2(k)]$$

Figure 51. Beam Angle, Beam Rate ($\theta, \dot{\theta}$)
Kalman Estimator.

derived as follows. It can be shown that the aircraft acceleration normal to the glideslope is approximately

$$\ddot{y} = V\dot{\Omega} \quad (3.15)$$

where V is the aircraft velocity and $\dot{\Omega}$ is the pitch rate of the aircraft, further,

$$R\ddot{\zeta} = \ddot{y} \quad (3.16)$$

where R is the range from the station and $\ddot{\zeta}$ is the angular acceleration with respect to the station.

Substituting the forcing term, equation 3.16 becomes

$$\ddot{\zeta} = V\dot{\Omega}/R. \quad (3.17)$$

Discretizing this equation yields

$$\dot{\zeta}_k = \dot{\zeta}_{k-1} + J V \dot{\Omega}/R_{k-1}. \quad (3.18)$$

Making the identification $X_2(k) = \dot{\zeta}_k$ completes the derivation of the glideslope Kalman estimator. The discussion of paragraph 3.2. relative to the use of the Q_k and R_k matrices to tailor the Kalman gains also applies.

3.3 Manual Heading Control Law

This mode is used to acquire and maintain a selected heading. Manual Heading mode is engaged by switching the Airspeed Controller mode switch to the MAN HDG position. The computational algorithm generates a steering needle command derived from heading error and bank angle. Bank angle command (PHIC) is gain programmed with true airspeed according to equation 3.19.

$$GHH = 2.0 + \text{limit } (1.0 (UO-350.0)/150.0)$$

$$PHIC = GHH + \text{limit } (30.00, \text{HEADING ERROR})$$

Figure 117 illustrates the Manual Heading.

3.4 TACAN Control Laws

The TACAN control laws are designed to capture and track TACAN radio information with provisions made to minimize overstation cone of confusion problems. Provisions are provided for automatic switching to a new outbound radial as the station is overflown. The sub-modes of TACAN are:

- 1) Heading Hold
- 2) Capture
- 3) Track
- 4) Overstation

The heading hold mode is included in the TACAN control laws to maintain a preset compass course. This preset compass course is flown until the aircraft has moved such that the capture conditions are satisfied. Once the capture conditions are satisfied, the mode of the DFDC is switched to capture. The gain on heading error is a programmed function of airspeed.

The capture mode is used to control the aircraft during the capture phase of TACAN mode until conditions for switching to track mode are satisfied. During the capture mode, the bank command will cause the aircraft to follow a piece-wise circular path.

The track mode is used to control the aircraft after the TACAN beam has been attained. The track mode generates a bank command which when implemented by the pilot will cause the aircraft to closely track zero beam deviation.

The overstation mode is a manual heading type mode which causes the aircraft to fly course datum error to zero.

A typical sequence of operation proceeds as follows. On engagement of TACAN mode the DFDC remains in heading hold until the capture conditions are satisfied. Capture is then engaged and flown until the track conditions are satisfied. The mode is switched from capture to track and a smooth transition to track occurs, resulting in the aircraft following the selected radial. At a pre-programmed distance (10.0nm at 500.0 knots) from the station, the DFDC switches to overstation mode. Overstation mode commands the aircraft to fly the course selected. At this point the pilot may select a new outbound radial by dialing up a new course selection. The system will then command the aircraft to the new course based on course datum error. On exit from the overstation mode, the aircraft will most probably be parallel to the selected radial, but offset from it. The digital flight director switches to capture mode until the new radial is acquired and then tracks it.

Flow diagrams presented in Figures 112 through 118 in Appendix A illustrate the complete documentation for TACAN mode.

3.4.1 Heading Hold Mode

The heading hold mode will be maintained once TACAN mode is selected on the air speed controller until the capture conditions are satisfied. Notice that the DFDC will switch directly into capture mode if TACAN is selected after the capture trip point has been satisfied. Basically the heading hold mode bank command (equation 3.19 and Figure 117) will maintain the aircraft on the heading

$$\begin{aligned} \text{GHH} &= 2.0 + \text{LIMIT} (1.0, (\text{UO} - 350.0)/150.0) \\ \text{PHIC} &= \text{GHH} * \text{LIMIT} (30.0^\circ, \text{HEADING ERROR}). \end{aligned} \quad (3.19)$$

selected as a function of heading error. The gain of heading error is a pre-programmed function of airspeed.

Capture mode engagement occurs when the aircraft reaches a computed capture trip point. There are two conditions that will cause capture to occur. These two conditions are 1) the LBS override is set or 2) the normal capture trip point is satisfied. During normal operation, LBS not set, there are three conditions that must be satisfied before the mode of operation is switched from heading hold to capture. These conditions are illustrated in equation 3.20 and Figure 112.

- 1) $ABS(RH \cdot QH) \leq 10.0NM$.
- 2) The Kalman Filter must have been running for 3.0 sec.
- 3) a) PHICL and PHICT must be of the same sign. (3.20)
 b) $PHICT \geq ADJPHL \cdot PHICL$
 where $ADJPHL = 0.35 + LIMIT(0.15, 0.3 \cdot (RH - 70.0) / 80.0)$

OR

- a) $ABS(QH) \leq 15 \mu A$.AND. $PHICT \leq 1^\circ$

3.4.2 Capture Mode

The capture mode has been designed such that it will cause the aircraft to fly a piece-wise circular path to intercept the selected TACAN radial. (For very small initial course errors an exponential type capture will result.) The computational algorithm is illustrated in equation 3.21 and Figures 113 and 117

$$\begin{aligned}
 GP &= 1.0 \\
 IF (RH \geq 50.0NM) GP &= 50.0/RH \\
 CF &= EXP(-JO/30) \\
 X2L &= CF \cdot (X2L - QDB) + QDB \\
 XSIL &= CF \cdot (XSIL - GAMH) + GAMH \\
 YDT &= X2L - VH \cdot (GAMH - XSIL) \quad (3.21) \\
 PHICT &= GP \cdot (KY \cdot QH \cdot RH + K\dot{Y} \cdot YDT) \\
 \text{where } KY &= 5.00^\circ/NM \\
 K\dot{Y} &= 0.12^\circ/KNOT \\
 PHICL &= ATAN(VH^2 \cdot (1.0 - \cos(GAMH)) / (G \cdot RH \cdot OH)) \\
 PHIC &= MIN(PHICL, PHICT)
 \end{aligned}$$

In addition, the capture mode insures that the system will properly handle a situation where a capture is initialized in a near parallel or fly away condition relative to the selected radial. Such a condition may well occur on exit from overstation mode. The track logic has been designed such that the circular command will be maintained until the beam deviation is sufficiently small and a smooth transition from capture to track will occur. Figures 113 and 114 illustrates the control logic for transition from capture to track.

3.4.3 Track Mode

In the track mode the aircraft is commanded to hold the selected radial with a bank command (equation 3.21, PHICT) that is a linear function of cross track deviation (CTD) and cross track deviation rate (CTR). The noise level on the TACAN beam data is greater than that on the localizer beam so that much more care is necessary in deriving CTR if bank activity is to be held to a minimum. A complementation of the Kalman filter rate and aircraft track angle has been formed to produce a cleaner rate signal in this mode. The track gains are set such that adequate tracking is maintained, but such that the bank activity is not excessive. As discussed earlier when the range from the station is less than a programmed trip point, the mode is

switched from track to overstation. Figure 115 illustrates the control logic for transition from track to overstation.

3.4.4 Overstation Mode

The overstation mode is used to command the aircraft while the aircraft is in the station cone of confusion. The bank command (equation 3.22 and Figure 117) is a product of a gain term

$$\begin{aligned} \text{GHH} &= 2.0 + \text{LIMIT}(1.0, (\text{UO} - 350.0)/150.0) \\ \text{PHIC} &= \text{GHH} * \text{LIMIT}(30.0^\circ, \text{COURSE ERROR}) \end{aligned} \quad (3.22)$$

which is a function of airspeed and course datum error. This permits the pilot to turn the aircraft parallel to a new outbound radial by turning the course selection knob when he is in overstation. Once the station has been overflowed and range again reaches the overstation trip point, the mode is switched from overstation to capture, initializing the acquisition of the new radial. Figure 116 illustrates the control logic for overstation to capture switching.

3.4.5 Range Estimation

Range estimation is not necessary in TACAN mode since DME range information is available. However, a DME range smoothing algorithm is required and is illustrated in Figure 119.

3.4.6 Wind Estimation

The lateral axis wind computational algorithm for the DFDC consists of separating the wind computation into an along track component and a cross track component, equation 3.23. The

$$\begin{aligned} \text{GAME} &= \text{GAMH} + T G \sin(\text{PHI}) / (\text{VHG} \cos(\text{PHI})) \\ \text{ATE} &= \text{RH} - \text{RHO} + T \text{VHG} \cos(\text{GAME}) \\ \text{CTE} &= \text{RH} \text{QH} - \text{RHO} \text{QHO} + T \text{VHG} \sin(\text{GAME}) \\ \text{ATW} &= \text{ATW} + \text{LA} \text{ATE} / T \\ \text{CTW} &= \text{CTW} + \text{LC} \text{CTW} / T \\ \text{WH} &= \sqrt{\text{ATW}^2 + \text{CTW}^2} \\ \text{OH} &= \text{ATAN}(\text{CTW} / \text{ATW}) \\ \text{CWH} &= \text{WH} \cos(\text{OH} - \text{PSI}) \\ \text{SWH} &= \text{WH} \sin(\text{OH} - \text{PSI}) \\ \text{VHG} &= \sqrt{\text{SWH}^2 + (\text{UO} + \text{CWH})^2} \\ \text{GAMH} &= \text{PSI} + \text{ATAN}(\text{SWH} / (\text{UO} + \text{CWH})) \end{aligned} \quad (3.23)$$

computational algorithm using along track and cross track wind, to compute wind, functioned properly and is illustrated in Figure 47.

However, from simulation results that illustrated that the along track wind component was not required, the along track wind component was dropped, resulting in the present form of the computational algorithm which is illustrated in equation 3.24.

$$\begin{aligned}
\text{GAME} &= \text{GAMH} + T G \sin(\text{PHI}) / (\text{VHG} \cos(\text{PHI})) \\
\text{CTE} &= \text{RH QHO} - \text{RH QH} - T \text{VHG} \sin(\text{GAME}) \\
\text{CTW} &= \text{CTW} + \text{LC LIMIT} (0.0012, \text{CTW}) / T \\
\text{WH} &= \text{ABS}(\text{CTW}) \\
\text{OH} &= \text{SIGN}(90.0^\circ, \text{CTW}) \\
\text{CWH} &= \text{WH} \cos(\text{OH} - \text{PSI}) \\
\text{SWH} &= \text{WH} \sin(\text{OH} - \text{PSI}) \\
\text{VHG} &= \sqrt{\text{SWH}^2 + (\text{UO} + \text{CWH})^2} \\
\text{GAMH} &= \text{PSI} + \text{ATAN}(\text{SWH} / (\text{UO} + \text{CWH}))
\end{aligned}
\tag{3.24}$$

3.5 ILS Localizer Control Laws

The localizer control laws are designed to capture and track ILS localizer radio information. The sub-modes of ILS localizer are:

- 1) Heading Hold
- 2) Capture
- 3) Track

The heading hold mode is included in the localizer control laws to maintain a preset compass course. This preset compass course is flown until the aircraft has moved such that the capture conditions are satisfied. Once the capture conditions are satisfied, the mode of the DFDC is changed to localizer capture. The gain on heading error is a programmed function of airspeed.

The capture mode controls the aircraft during the capture phase of localizer mode until conditions for switching to track mode are satisfied. During the capture mode, the bank command will cause the aircraft to follow a piece-wise circular path.

The track mode controls the aircraft after the localizer beam has been acquired. The transition from capture mode to track mode consists of fading the bank command from one control to the other. The track mode generates a bank command which when implemented by the pilot will cause the aircraft to closely track zero beam deviation.

A typical sequence of operation proceeds as follows. On the selection of ILS mode on the air speed controller, the DFDC would remain in heading hold mode until the capture conditions are satisfied. Once the capture conditions are satisfied, the capture mode would be engaged and flown until the track conditions are satisfied. Having satisfied the track conditions, the track mode will be engaged and a smooth transition in bank command will occur. Once having engaged the track mode, the DFDC will remain in the track mode until a new mode is selected on the air speed controller.

Flow diagrams presented in Figures 118, 120, 121, and 122 in Appendix A illustrate the complete documentation of ILS localizer mode.

3.5.1 Heading Hold Mode

The heading hold mode provides a bank command (equation 3.25 and Figure 118) which when implemented will

$$\begin{aligned} GHH &= 2.0 + \text{LIMIT}(1.0, (UO - 350.0)/150.0) \\ PHIC &= GHH * \text{LIMIT}(30.0^\circ, \text{HEADING ERROR}) \end{aligned} \quad (3.25)$$

Maintain the aircraft on a preset compass heading. The gain on heading error is programmed function of airspeed.

The heading hold mode will be maintained once ILS mode is selected on the air speed controller until capture conditions are satisfied. Capture mode engagement occurs when the aircraft reaches a computed capture trip point. There are two conditions that will cause the mode to be switched to the capture mode. These two conditions are either 1) the LBS override is set or 2) the normal capture trip point is satisfied. These conditions are illustrated in Figure 120. During normal operation, LBS is not set, there are three conditions that must be satisfied before the mode of operation can be switched from heading hold to capture mode. These conditions are as follows:

- 1) The estimated radio deviation must be less than 240uA for 3.0 sec. (This will prevent false captures.)
- 2) The calculated capture bank command must be greater than $(2.0^\circ + \text{ABS}(\text{estimated radio deviation})^\circ)$.
- 3) RANGE .GT. 10.0 NM: The estimated radio deviation must be Less than 180 uA.

RANGE .LT. 10.0 NM: The estimated radio deviation must be less than $180\text{uA} + 6.0 (10.0 - \text{RH})\text{uA}$.

When the aircraft reaches the computed capture trip point, the DFDC control law mode is switched from heading hold mode to capture mode. The mode of the digital flight director will also be switched from heading hold mode to the capture mode if:

- 1) The estimated beam deviation has been less than 240uA for 3.0 sec.
- 2) The estimated beam magnitude is less than 30uA.

3.5.2 Capture Mode

The capture mode controls the aircraft from the calculated capture trip point until the aircraft has moved within the track trip point. The computational algorithm (equation 3.26 and Figure 118 and 122) for

$$PHICL = \text{ATAN}(VHG^{**2} * (1.0 - \text{COS}(GAMH)) / (G * RH * QH)) \quad (3.26)$$

the capture mode bank command will cause the aircraft to follow a piece-wise circular path until track mode is engaged. Figure 121 illustrates the four conditions that will cause the track mode to be engaged. These conditions are as follows:

- 1) The estimated radio deviation is less than 30.0 uA.
- 2) The estimated radio deviation is less than ABS (45.0 uA) and the estimated gamma is less than 20.0°.
- 3) The estimated radio deviation is less than -45.0 uA and the estimated gamma is greater than 20.0°.
- 4) The estimated radio deviation is greater than 45.0 uA and the estimated gamma is less than -20.0°.

When one of the above conditions is satisfied, the DFDC mode is switched from capture to track.

3.5.3 Track Mode

The track mode is the final mode of the localizer algorithm and provides a bank command that is a linear function of y and \dot{y} . The track mode provides a smooth transition from the capture mode to the track mode. At the instant of transfer from the capture mode, a fader command value is retained. The retained fader command value is the bank command difference between the capture mode and the track mode laws. During the initial part of the track mode the fader command is geometrically decreased to zero.

When the computed track bank command (equation 3.27 and Figure 122)

$$\begin{aligned}
 GP &= 8.0 / (7.0 + 573.0 \cdot \text{ABS}(QH)) \\
 PHICT &= GP \cdot KY \cdot QH + KY \cdot QDB \\
 \text{IF } (RH \text{ .LT. } 5.0\text{NM}) \text{ PHICT} &= RH \cdot PHICT / 5.0 \\
 \text{WHERE } KY &= 4.0^\circ / 70.0 \text{ FT} \\
 K\dot{Y} &= 35.0^\circ / 70.0 \text{ FT/SEC.}
 \end{aligned} \tag{3.27}$$

is implemented, the aircraft closely tracks zero estimated beam deviation.

3.5.4 Range Estimation

The computational algorithm for the localizer range estimate is illustrated in equation 3.28 and Figure 122 . A range measurement is computed using the

$$\begin{aligned}
 RA &= VGH \cdot \sin(QH - GAMH) / QDB \\
 RE &= RH - T \cdot VHG \cdot \cos(GAMH) \\
 \text{IF } (\text{ABS}(QH) \text{ .LT. } 75\text{UA}) \text{ .AND.} \\
 &\quad (\text{LATSW .NE. MANUAL HEADING}) \text{ AY} = 0.95 \cdot \text{AY} \\
 RH &= RE + AY \cdot \text{LIMIT} (20.0 \cdot JO, RA - RE) \\
 \text{IF } (RH \text{ .LT. } 5.0\text{NM}) \text{ RH} &= 5.0\text{NM} \\
 \text{IF } (GSCORT) \text{ RH} &= RE + 0.1 \cdot (SH - RE + 1.5\text{NM})
 \end{aligned} \tag{3.28}$$

Kalman derived radio rate. The range is then smoothed through a predictor-correlator type filter.

Once glideslope has engaged, RH is up-dated as a function of radar altitude.

3.5.5 Wind Estimation

The discussion of the lateral wind computation was covered in paragraph 3.4.6.

3.6 ILS Glideslope Control Laws

The ILS glideslope control laws are designed to capture and track ILS glideslope radio information. The sub-modes of ILS glideslope are:

- 1) Idle
- 2) Precapture
- 3) Capture
- 4) Track

The idle mode is a do-nothing mode. That is, no computations are performed except to check if the Kalman filter should be started. Once the conditions to start the Kalman filter are satisfied, the DFDC mode is switched from idle to precapture. However, in the simulation results, a pitch command is computed to maintain a preset altitude.

The precapture mode performs all computations that are in the glideslope algorithm. In addition, it computes the glideslope capture trip point. Once the capture trip point has been satisfied, the DFDC is switched from precapture to capture mode.

The capture mode controls the aircraft during the capture phase. The computed pitch command will cause the aircraft to follow a piece-wise circular path. The capture mode is maintained until conditions are satisfied for glideslope track engage, at which time the mode is switched from capture to track.

The track mode is the final glideslope mode and calculates a pitch command which when implemented will cause the aircraft to closely track zero beam deviation.

During the idle and pre-capture modes, the horizontal steering pointer, HPTR, is biased out of view. HPTR is brought in view only at glideslope capture.

A typical sequence of operation proceeds as follows. On selection of the ILS mode on the air speed controller, the DFDC would remain in the idle mode until localizer no longer was in heading hold mode and the glideslope beam deviation was less than 120.0UA. Once these two conditions are satisfied, the mode would be switched to pre-capture mode and the Kalman filter would be initialized. The pre-capture mode would be maintained until the capture conditions are satisfied, at which time the mode would be switched from pre-capture to capture. Capture would be flown until the track conditions are satisfied. Having satisfied the track conditions, the track mode would be engaged and a smooth transition in pitch command would occur. Once having engaged the track mode, the DFDC will remain in the track mode until a new mode is selected on the air speed controller.

Flow diagrams presented in Figures 123, 124, 125, 126, and 127 of appendix A illustrate the complete documentation of ILS glideslope mode.

3.6.1 Precapture and Idle Modes

The purpose of the idle mode is to compute when to start the glideslope Kalman filter and the glideslope algorithm. This occurs when the glideslope deviation is less than 120.0uA and the localizer mode is not in manual heading, at which time the glideslope mode is switched from idle to precapture.

The precapture mode performs all computations of the glideslope algorithm and computes the capture trip point. The transition from capture to track will occur when either 1) the VBS override is set or 2) the estimated beam deviation is less than 35.0uA.

Figures 123, 124, and 126 illustrate the idle and pre-capture modes, respectively.

3.6.2 Capture Mode

The capture mode controls the aircraft during the capture phase of glideslope. The computational algorithm (equation 3.29 and Figures 126 and 127) for the capture mode pitch command will cause

$$\begin{aligned} \text{GDF} &= \text{EXP}(-\text{JO}/2.0) * (\text{GDF} - \text{GDH}) + \text{GDH} \\ \text{ATGSN} &= \text{GDF} * \text{SH} / (\text{UO} + \text{ATSW}) \\ \text{RPN} &= \text{SH} * \text{GH} / (1.0 - \text{COS}(\text{ATGSN})) \\ \text{TDC} &= -\text{VHTN} / (3600.0 * \text{SIGN}(\text{RPN}, \text{ATGSN})) \\ \text{TC} &= \text{TC} + \text{JO} * \text{TDC} \\ \text{HPTR} &= 3.0 * (\text{TC} - \text{TM}) - 0.02 * \text{UF2} \end{aligned} \quad (3.29)$$

the aircraft to follow a piece-wise circular path until the track mode is engaged. The track mode is engaged once the estimated glideslope deviation becomes less than 15.0uA. Figure 125 illustrates the capture to track control logic.

3.6.3 Track Mode

The track mode is the final mode of the glideslope algorithm and provides a pitch command (equation 3.30 and Figures 126 and 127) that is a linear function of y and \dot{y} . When the track pitch

$$\begin{aligned} \text{TF15} &= \text{EXP}(-\text{JO}/15.0) * (\text{TF15} - \text{TM}) + \text{TM} \\ \text{TF15C} &= \text{TM} - \text{TF15} \\ \text{AED1} &= \text{KY} * \text{GH} * (1.0 - \text{VHTN}/360.0) \\ \text{AED1} &= \text{AED1} + \text{KY} * (0.25 * \text{GDF} + 0.75 * \text{GDH}) / \text{VHTN} \\ \text{HPTR} &= \text{SH} * \text{AED1} - \text{TF15C} \\ \text{WHERE } \text{KY} &= 1^\circ/10 \text{ FT} \\ \text{KY} &= 1.5^\circ/10 \text{ FT/SEC} \\ \text{TF15C} &= \text{PITCH THROUGH A 15 SEC. FILTER} \end{aligned} \quad (3.30)$$

command is implemented, the aircraft will closely track zero estimated beam deviation.

3.6.4 Range Estimation

The computational algorithm for the range estimation is illustrated in equation 3.31 and Figure 109. A range measurement is computed from the

$$\begin{aligned}
SA &= Z/(2.50 - GH) \\
SE &= SH - T \cdot VHTN \cdot \cos(ATGSN) \\
SH &= 0.9 \cdot SE + 0.1 \cdot SA
\end{aligned}
\tag{3.31}$$

Aircraft radar altitude. The computed range is then smoothed through a predictor-correlator type filter.

3.6.5 Wind Estimation

The glideslope wind computational algorithm for the DFDC follows directly from the lateral axis wind algorithm. The computational algorithm is based on computing along track wind only. Equation 3.32 and Figure 127 illustrates the computational algorithm and the glideslope wind estimation.

$$\begin{aligned}
TF1 &= \exp(-JO) \cdot (TF1 - TM) + TM \\
TDM &= TM - TF1 \\
ATGSE &= ATGSN + JO \cdot TDM \\
ATSE &= SH - SHO + T \cdot VHTN \cdot \cos(ATGSE) \\
ATSW &= ATSW - 0.008 \cdot ATSE/T \\
VHTN &= UO + ATSW
\end{aligned}
\tag{3.32}$$

SECTION IV

4.0 Software Design

This section of the report presents an indepth discussion of the software. The Fortran programs were the design median for the DFDC software. The AED programs were used for software definition except where the AED language could not express the desired result. The final result of both software program descriptions is the assembly program description which is the DFDC software. The following five modes:

- 1) Self Test
- 2) Off
- 3) Manual Heading
- 4) TACAN
- 5) ILS

are described in detail in paragraphs 4.1, 4.2, 4.3, 4.4, and 4.5, respectively. These five modes are each selectable via the air speed controller. Paragraph 4.6 contains a discussion on mode control. Paragraph 4.7 contains a discussion of how the Kalman estimator was implemented. Paragraph 4.8 describes the math package required for the system.

The system specification of the software function was developed via hybrid simulation in Fortran IV using floating point arithmetic. At completion of the hybrid simulation the system description was converted to a detailed AED description of the system software. The AED description was then hand compiled into a machine level symbolic assembly language program for the CAPS (Collins Adaptive Processor System) processor being used as the digital flight director processor. This program was assembled and partially tested using the CAPS assembler/simulator. Additionally, the AED system description was tested by 1108 simulation. The final program debug was performed using the digital flight director static test bench and the hybrid simulator.

In addition, hybrid simulation was also performed in Fortran IV using scaled fraction arithmetic, to verify the scaling effort.

The system software has been structured such that there are three stacks implemented in the machine code. These stacks are as follows:

- 1) Interrupt Stack
- 2) Lateral Axis Stack
- 3) Longitudinal Axis Stack

The interrupt stack handles all I/O processes. The lateral axis stack consists of Off Mode, Self Test, Manual Heading, TACAN, and Localizer computations. The longitudinal axis stack consists of glideslope only. The time required to complete the computational algorithms for each mode is as follows:

STM, OFF - 5%

Manual Heading - 15%

TACAN - 50%

ILS - 80%

Where the % indicates the fraction of the tenth of a second allowed for the computation of the indicated function. For instance, manual heading mode requires only 15 milliseconds to convert the required inputs, compute a bank command, output the command, and return to the wait loop.

Figure 129 illustrates the ILS mode timing; Figure 128 illustrates all other mode timing. The basic computational cycle begins every 100 millisecond, a sample rate of 10 times per second. A timed interrupt initiates the process. After sampling the required inputs for the computation, the program enters a mode control sub-routine to determine which computation is to be performed (the modes are illustrated in Figures 109 and 130). Control is then transferred to the proper computational routine which computes and updates output values. Upon completion of the computation the program rests in a wait loop until interrupted for the next cycle. To avoid large delays in the glideslope computation (between input sample time and output update time), a 20-per-second timer is used. The glideslope computation is performed in the second half of the 100 millisecond cycle, if the DFDC is in the ILS mode.

If an illegal address occurs during operation, machine generated interrupts will cause the fail warning sub-routine to be entered. This sub-routine will set FD VALID invalid and set both pointer flags in view. If other error conditions occur such as illegal opcode, stack overflow, and others, the machine will stop.

Software costs are minimized by use of a high level language to describe the problem and to program the solution. This outside-in software engineering approach provides a processing structure within which variations are economically produced.

A stack-oriented machine architecture allows efficient compilation from the high-level language to the machine code, minimizing program storage space. Even if machine level programming is done, the programmer is relieved of most of the addressing and memory allocation problems inherent in register oriented machines.

4.1 Self Test Mode

Self test mode can only be initiated when the air speed controller is in the off mode. The initiation of the self test mode either by switching to self test on the air speed controller or pushing the front panel push button will cause the DFDC to perform internal tests which evaluate the units integrity. These tests are:

- 1) Deflection of the VPTR and HPTR one quarter of an inch to the right and one quarter of an inch down. The outputs are forced to this position by the computer D/A outputs. These output signals are also multiplexed to the A/D converter. Digitized and interrogated for positional accuracy.
- 2) The self test constant is requested from the A/D and compared with constants stored within the computer to test the I/O control logic, analog multiplexer, and the digital flight director bus system.
- 3) The execution of basic arithmetic and reference/assign operations.
- 4) Program sequencing.

These functions are illustrated in Figure 110

4.2 Off Mode

Off mode can only be initiated by selecting off mode on the air speed controller. When the DFDC is in any mode other than the self test mode certain monitor functions are performed:

- 1) A check is performed to determine that the computations have been completed in the time allotted: DONE has been set.
- 2) The occurrence of an invalid address or opcode.

Incorrect operation during any of these tests results in actuation of the front panel failure annunciator and a high impedance state at the DFDC valid output - FD VALID light is off.

When the DFDC is in the off mode, the HPTR and VPTR are biased out of view, down and to the right.

These functions are illustrated in Figure 109.

4.3 Manual Heading Mode

Manual heading mode is engaged by switching the air speed controller to manual heading. The computational algorithm generates a steering needle command consisting of manual heading error times a gain term which is a programmed function of air speed less bank angle.

The software implementation of the manual heading mode follows directly from the discussion for TACAN manual heading design.

This function is illustrated in Figure 111.

4.4 TACAN mode

TACAN mode is engaged by switching the air speed controller to TACAN. The computational algorithm generates a steering needle command which will 1) maintain the selected heading until capture is engaged, 2) capture the TACAN radial via a steering needle command that will cause a piece-wise circular arc to be flown, 3) adequately track the TACAN radial, and 4) maintain the selected TACAN radial during the overstation mode.

The software implementation of TACAN follows directly from the discussion in the TACAN design section.

These functions are illustrated in Figures 112 through 118.

4.5 ILS Mode

The ILS mode is engaged by switching the air speed controller to ILS. The computational algorithm for ILS mode generates two steering needle commands which are HPTR and VPTR. HPTR displays a command which when implemented will cause the aircraft to capture and track the glideslope beam. VPTR displays a command which when implemented will 1) maintain the selected heading until capture is engaged, 2) capture the localizer beam via a steering needle command that will cause a piece-wise circular arc to be flown, and 3) adequately track the localizer beam.

The software implementation of ILS follows directly from the discussion on the ILS design section.

These functions are illustrated in Figures 118 through 127.

4.6 Mode Control

Mode control is illustrated in Figures 109 and 130.

When the air speed controller is switched from one mode to another mode, the processor enters the off mode, executes a re-start, then enters the selected mode.

4.7

Kalman Filter

Since the Kalman filter can be described by four matrix equations, the initial preference for coding it was to implement a generalized Kalman filter program based on matrix arithmetic sub-routines (transpose, add, multiply, etc.). This implementation would have resulted in a very slow computational algorithm as well as using more memory than a "customized" approach would have required. It was, therefore, decided to "customize" the Kalman filter as follows. A software sub-routine was designed to perform the specific operation transpose ($TRANS \times M$), where TRANS is the transition matrix and M is some other 4×4 matrix. This sub-routine is called twice per each of the three Kalman filters, and its use resulted in a reduction of the filter calculations memory requirements by 30%. The over-all Kalman calculations require about 15 milliseconds.

The Kalman filter was coded using double precision arithmetic instructions.

The Kalman filter algorithm has been divided into two parts. These two sub-routines are 1) the Kalman gain algorithm and 2) the positional vector update algorithm. This division allows more efficient code generation and allows the program to be structured easier. These two sub-routines are named KFIL1 and KFIL2, respectively.

4.8

Math Package

In addition to the machine arithmetic instructions discussed in Section III, the following functions have been implemented:

- 1) ABSOLUTE VALUE
- 2) ARC TANGENT
- 3) COSINE
- 4) LIMIT
- 5) SIGN
- 6) SINE
- 7) SQUARE ROOT

All functions are 16 bit computations. ABSOLUTE VALUE, LIMIT, and SIGN are implemented in microcode. The other functions have been implemented by sub-routines. The computational algorithms used are described briefly below:

ABSOLUTE VALUE (X)

```
BEGIN
  ABS(X) = IF X GEQ 0.0 THEN X ELSE -X $.
  GOTO RETURN $.
END
```

ARC TANGENT (X)

0 .LE. X .LE. 1.0

BEGIN

X2 = X**2

ATAN = X*(B0 + $\frac{A1}{(X2 + B1 - \frac{A2}{(X2 + B2 - \frac{A3}{(X2 + B3))})})$))

GOTO RETURN \$.

END

WHERE

B0 = 0.174655438

A1 = 3.709256262

B1 = 6.762139240

A2 = 7.106760045

B2 = 3.316335425

A3 = 0.264768920

B3 = 1.448631538

COSINE (X)

BEGIN

COS = SIN(PI/2 + X) \$.

GOTO RETURN \$.

END

LIMIT (L. X)

BEGIN

LIMIT = IF (ABS(X) .GT. L)

THEN SIGN(L. X)

ELSE X \$.

GOTO RETURN \$.

END

SIGN(X. S)

BEGIN

SIGN = MAGNITUDE OF X WITH SIGN OF S \$.

GOTO RETURN \$.

END

SINE (X)

BEGIN

IF (PI/2 GRT ABS(X)) THEN GOTO 2 \$.

IF (X EQL -PI/2) THEN GOTO 1 \$.

X = SIGN (1.0 - ABS(X), X) \$.

IF (X EQL 1.0) THEN GOTO 2 \$.

1 X = 0.999998*X

2 X = X + X \$.

X2 = X**2 \$.

AED1 = (A3 + A4*X2) * X2 \$.

AED1 = (A3 + A4*X2) * X2 \$.

AED1 = (A2 + 0.1*AED1) * X2 \$.

AED1 = (A1 + 0.1*AED1) * X2 \$.

SIN = X * (A0 + AED1) + X \$.

GOTO RETURN \$.

END

WHERE

A0 = 0.1570796290

A1 = -0.6459633598

A2 = 0.7968848053

A3 = -0.4672227919

A4 = 0.0150820562

SQUARE ROOT (X)

BEGIN

IF X LES 0 THEN

BEGIN

SQRT = 0 \$.

GOTO RETURN \$.

END \$.

SQRTBI\$ Y = 0.5 * (Y + X/Y) \$.

N N-1 N-1

IF Y LES Y THEN GOTO SQRBI \$.

N-1 N

SQRT = Y

N

GOTO RETURN \$.

END

SECTION V

5.0 Simulation Studies

Section 5.0 provides a summary of the simulation results performed during the development and support phases of the DFDC program. Paragraph 5.1 presents the Kalman estimator simulation results. Paragraph 5.2, 5.3, and 5.4 contain the simulation results for manual heading, TACAN, and ILS modes, respectively.

Figure 52 illustrates the EAI PACER 100 digital computing system and its associated peripheral equipment. This system consists of a mag, tape operation system with a 16 bit, 32K word memory and a hardware floating point processor. The peripheral equipment consists of a CRT-display with keyboard and hard copy unit, a card reader, and a digital plotter. The on-site availability of the digital plotter adds significantly to the ability to process and present digital data in a readily understandable form.

Figure 53 illustrates the digital and analog computers combined with an interface unit between them. The interface unit has A to D and D to A conversion equipment necessary for high speed data transfer. It also has logic and control circuits which set up and check out the analog computer. This set-up and check-out capability of the analog computer by the digital insures accurate simulations from one set-up to the next. This is an important feature when large volume and fast turn-around times are needed as in the case for the simulation facility.

The cockpit illustrated in Figure 54 is tied to the hybrid system through an interface unit which converts DC voltages to required low level DC voltages, 2-wire AC voltages and 3-wire AC synchro voltages, and is used to drive the ADI, and HSI flight instruments. Information from the cockpit column and rudder pedals is fed back to the hybrid system. This communication between the hybrid system and the cockpit simulator provided the means to put a pilot-in-the-loop for hands-on evaluation of the DFDC flight control system.

Figure 55 illustrates the implementation of the DFDC hybrid simulation in the simulation facility. The PACER 100 digital computer modeled the DFDC, while the analog computer modeled the aircraft and the real world environment. Preliminary simulation checkout was facilitated by closed loop control with a simple pilot model. However, the cockpit tie into the simulation provided the final check-out of the man-in-the-loop for the DFDC. The simulation was digitally controlled via the graphics unit and the key board input.

Appendix A illustrates the analog simulation diagrams for the composite aircraft. Each aircraft simulated in this study was set from this analog simulation diagram by changing potentiometer settings. Potentiometers were set automatically from IBM cards read by the hybrid system. The hybrid was also used to perform aircraft checkout by setting up check traces under program control. With this technique, less than ten minutes

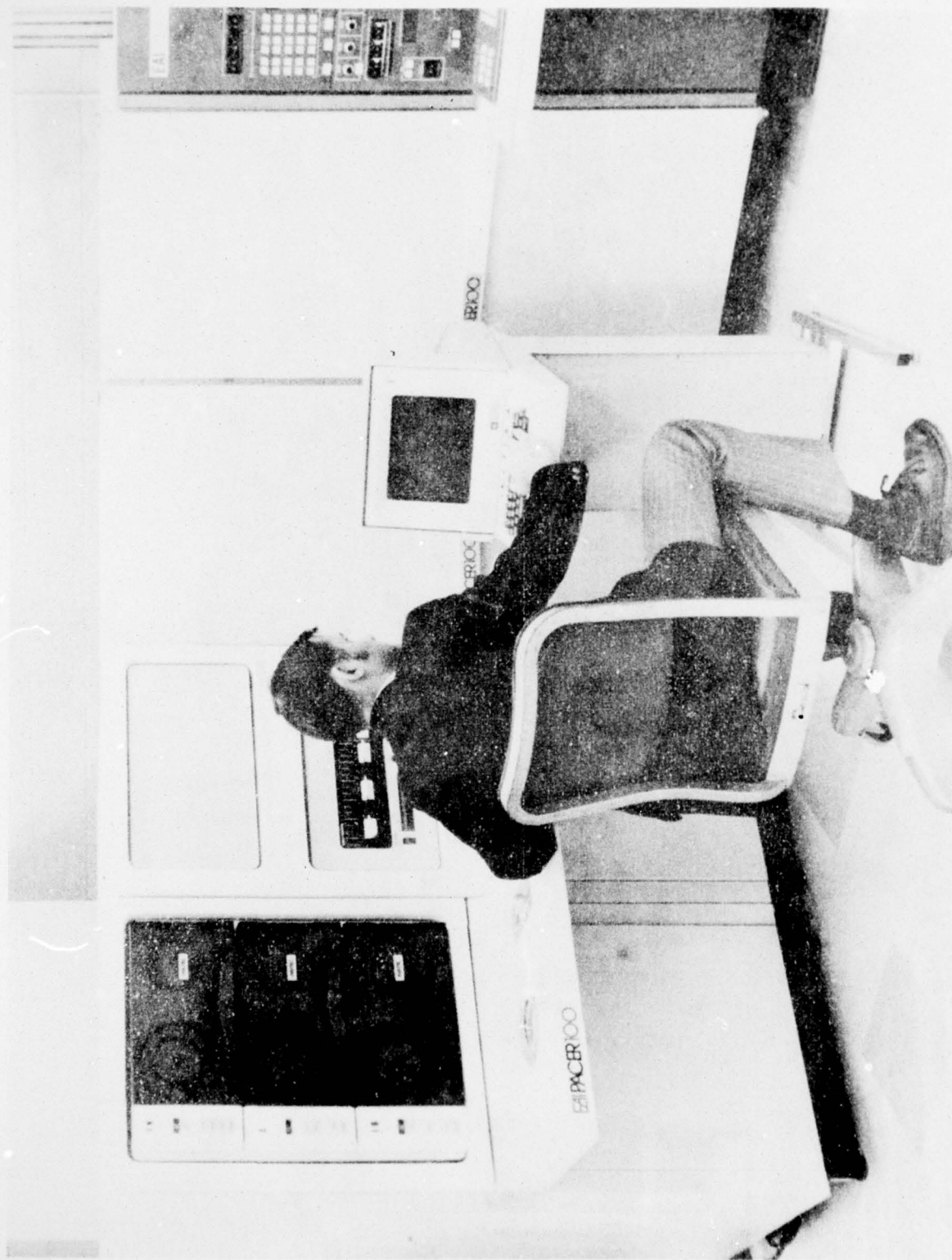


Figure 52. EAI PACER 100 Digital Computer and Peripheral Equipment

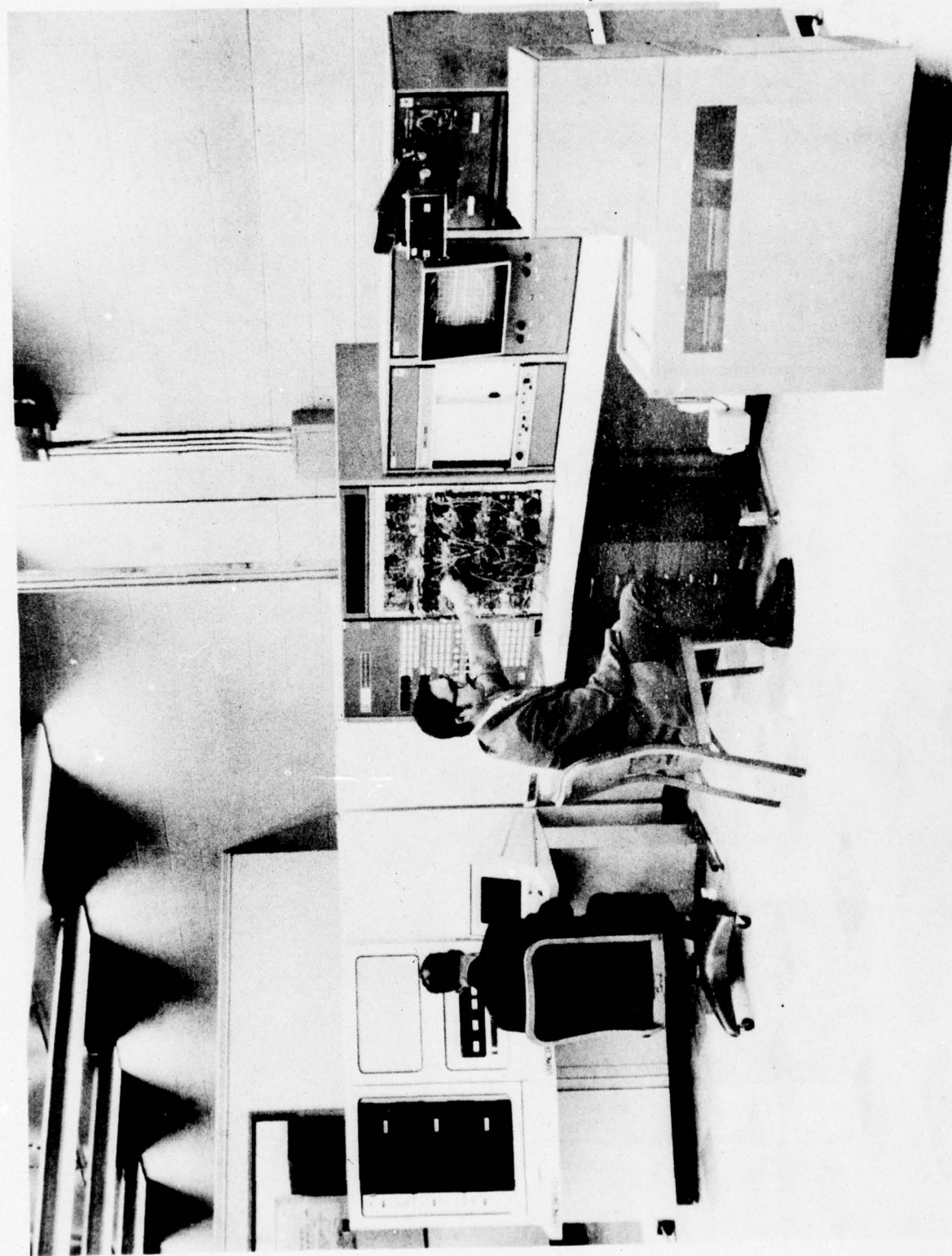


Figure 53.EAI PACER 100 Digital and 680 Computers and Interface Unit

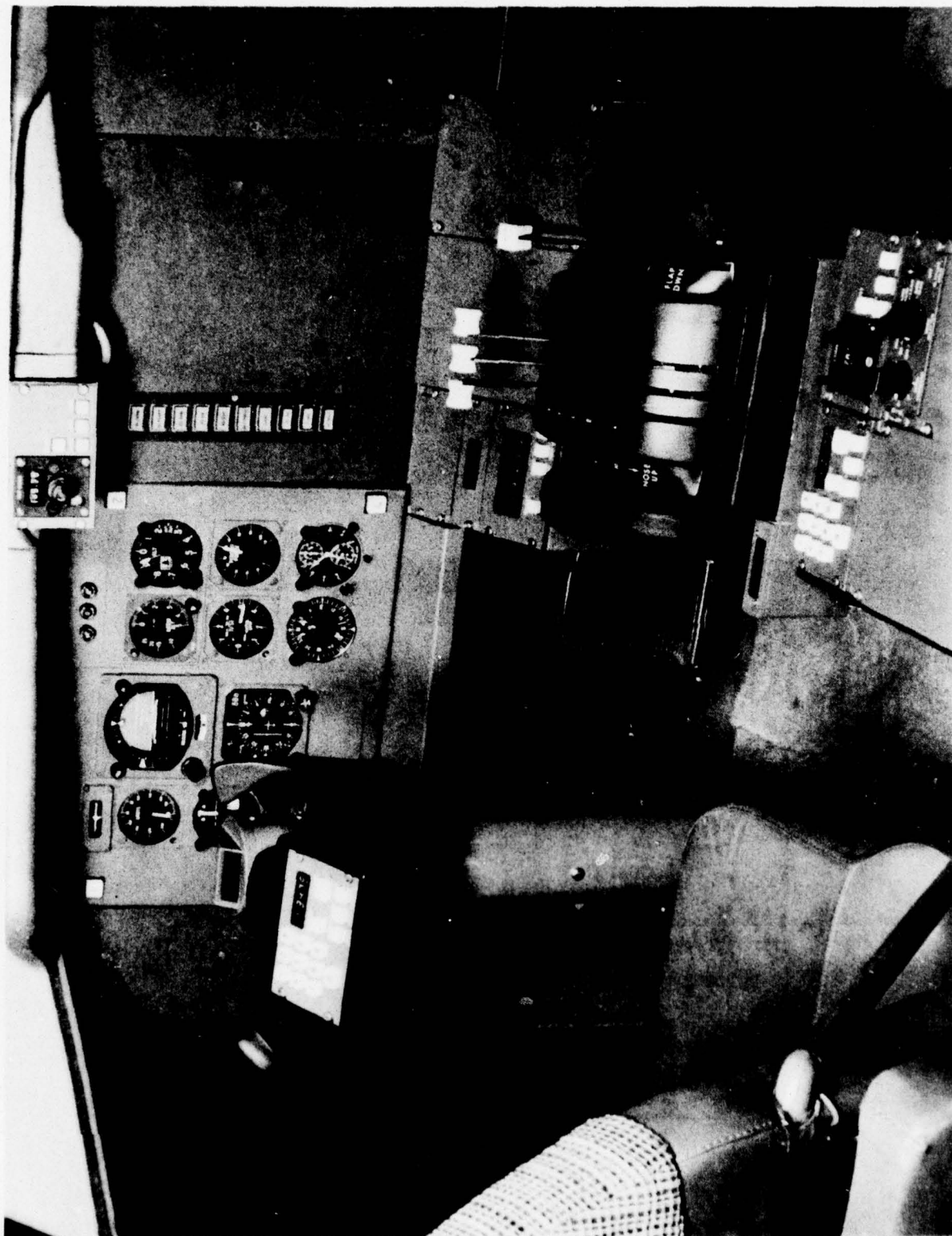


FIGURE 54. Cockpit Facility

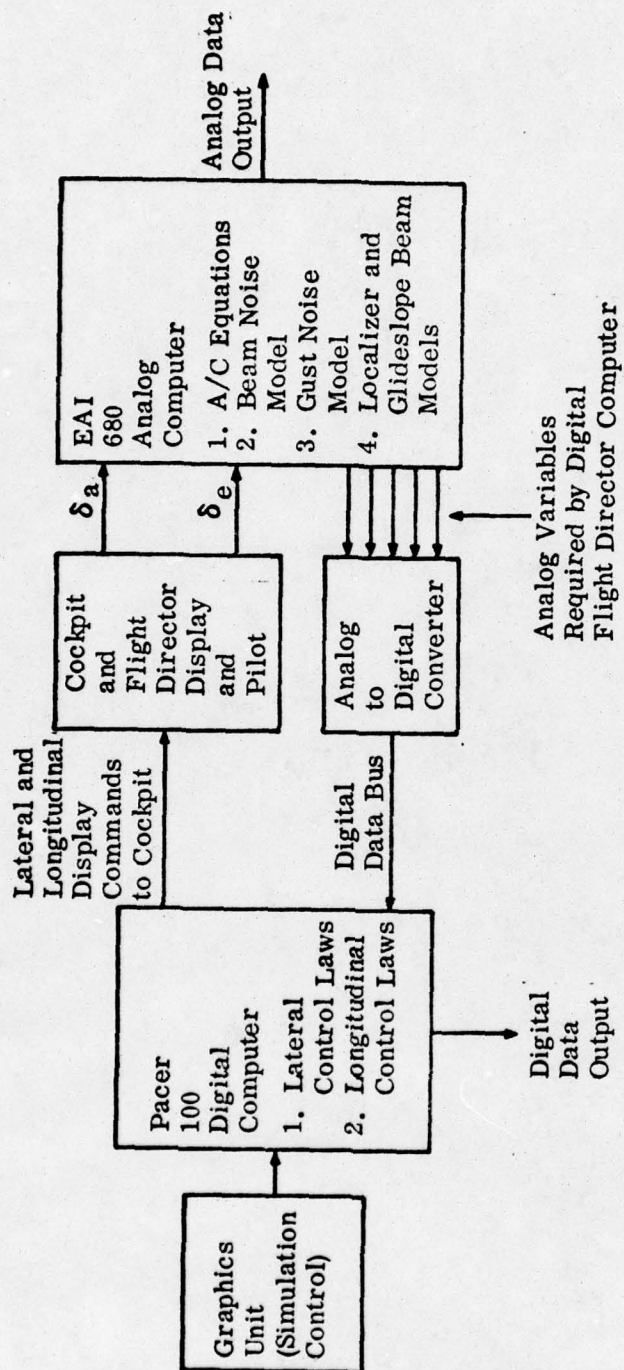


Figure 55. Digital - Analog Interface for Digital Flight Director Computer Hybrid Simulation.

was required to convert the simulation from one aircraft to another. The standard aircraft distance and angle definitions are illustrated in Figure 37. The basic geometric relationships for ILS localizer and glideslope are illustrated in Figures 38 and 39, respectively. Appendix B contains the definition of system variables as they were implemented in the computer programs.

One of the objectives of the DFDC program was to develop computations which were relatively insensitive to the aircraft dynamics. In order to study this aspect of the simulation problem, three widely differing aircraft were selected from our aircraft data library to cover the broad spectrum of possible future system installations. These aircraft included a jet transport, a propeller powered aircraft, and a high performance jet aircraft. The jet transport is typical of a large number of heavy, swept wing, jet aircraft that are in use both in civilian and military applications. This aircraft closely resembles the KC-135 tanker aircraft which the Air Force utilizes extensively. The propeller power aircraft was chosen since it well represents the large population of slower, straight wing aircraft. The jet aircraft represents the light swept wing, high performance aircraft end of the potential user spectrum. This type of aircraft is used in considerable quantity by the Air Force.

5.1 Kalman Estimator

The hybrid simulation results of the Kalman estimator performance are illustrated in Figures 56 through 62. The relationship of these figures is illustrated in the following:

FIGURE	BEAM NOISE	R_k
56A, 58	zero	$10E-19$
56B, 57A, 59	2.5uA	$10E-19$
56C, 57B, 60	4.81uA	$10E-19$
56D, 57C, 61	4.55uA	$10E-7$
56E, 57D, 62	5.59uA	$10E-6$

Figure 56 illustrates the capture and track of five ILS localizer approaches for different beam noise levels and values of R_k . Figure 57 illustrates the data sequence of the four sampled noise beams. Figures 58, 59, 60, 61, and 62 illustrate the dynamics of the Kalman gains, K , and the data sequence of the state vector, X_k . The intent of these simulation results is to illustrate the effect of varying R_k , the measurement noise covariance matrix, has on the Kalman estimator and its response. R_k provides the estimator with information relative to the signal to noise ratio of the incoming beam data. As R_k is increased, the Kalman gains, K_k , are decreased. The result of increasing R_k and decreasing K_k is illustrated in Figures 56 through 62. Examining the figures will demonstrate that increasing R_k from zero removes the high frequency noise content from the estimated beam, beam rate, and HPTR, and the Kalman gains decrease in value. The particular values of R_k that were selected for each mode were presented in section 3.0.

5.2

Manual Heading

The hybrid simulation results for manual heading mode are illustrated in Figures 63 through 66. These hybrid simulation results demonstrate satisfactory performance of the manual heading mode of the DFDC for the three aircraft at three different airspeeds. Each aircraft was flown in a disturbance free environment for a manual heading of zero degrees to $\pm 50^\circ$ and for zero heading error containing β gust. On close examination of these simulation results, it is observed that

- 1) For the disturbance free case, β equaled zero, the aircraft turned to the selected heading radial and acquired the radial with little or no overshoot.
- 2) For the disturbance case, β g not zero. The aircraft tracked zero heading error with satisfactory performance.
- 3) At 200 knots or less, PHIC = heading error.
The aircraft will be commanded to roll off the bank limit of 30° at a heading error of 30° .
- 4) At 350 knots, PHIC = $2 \times (\text{heading error})$.
The aircraft will be commanded to roll off the bank limit at a heading error of 15° .
- 5) At 500 knots or more, PHIC = $3 \times (\text{heading error})$.
The aircraft will be commanded to roll off the bank limit at a heading error of 10° .
- 6) The rate limit on bank command is $5^\circ/\text{second}$.

Figures 63, 64, 65, and 66 demonstrate the satisfactory performance of the manual heading mode for the DFDC for the jet transport aircraft flying at 496 knots, the jet transport aircraft flying at 125 knots. The propeller powered aircraft flying at 120 knots and the jet aircraft flying at 150 knots, respectively.

5.3

TACAN

The heading hold, capture, track, and overstation performance of the DFDC's TACAN modes are illustrated in Figures 67 through 73 for the jet transport flying at 495 knots. These simulation results demonstrate the satisfactory performance of TACAN mode. The TACAN radio beam was captured with a typical overshoot of less than $10\mu\text{A}$. The track mode maintained the radio beam less than $15\mu\text{A}$, typically.

Perturbations of course datum and distance-to-station are illustrated in Figures 67 through 70. These simulation results demonstrate the satisfactory performance of TACAN mode as follows:

- 1) The bank flown during the capture maneuver was essentially a constant value.

- 2) The beam overshoot was less than 15uA for the non-geometry limited case.

The effects of a 50 knot wind and 100 knot wind are illustrated in figures 71 and 72 , respectively. The wind computation in these hybrid simulations was removed. The results illustrate that the wind computation was not needed for TACAN mode, and has been removed.

The effects of beam noise are presented in Figure 73 . These results illustrate the satisfactory performance of TACAN mode.

5.4

ILS

All hybrid simulation results presented in this section have used the (0, 0) Kalman estimator to provide beam and beam rate information to the computational algorithms. Figures 74 through 92 demonstrate the satisfactory performance of ILS localizer manual heading, capture, and track modes. The satisfactory performance of the ILS localizer mode was demonstrated by the above figures as follows:

- 1) The bank flown during the capture phase of the maneuver was essentially a constant value except for geometry limited cases.
- 2) The estimated range was typically within 2NM of the correct value.
- 3) The estimated wind was typically less than 5 knots for no wind.
- 4) The circular capture algorithm was adaptive to course datum and distance-to-station variations as demonstrated by the figures in maintaining the capture overshoot to typically less than 15uA for the non-geometry limited case.
- 5) The circular capture algorithm was adaptive to airspeed changes as demonstrated by Figure 90 in maintaining the capture overshoot to typically less than 15uA for the non-geometry limited case.

The satisfactory performance of the ILS glideslope capture and track modes was demonstrated by Figures 93 through 95 . These simulation results demonstrate that the aircraft was command to capture and track the glideslope beam. The track mode maintained the aircraft typically within 10 ft. of the beam for wind and beam noise.

5.4.1 ILS Localizer

5.4.1.1 Jet Transport - 124.5 knots

Figures 74 through 77 illustrate the performance of the DFDC ILS localizer algorithm for heading hold, capture, and track modes for the

ILS localizer with a jet transport aircraft flying at 125 knots for 5, 10, 15, and 20 NM, respectively from threshold (glideslope antenna). These hybrid simulation results illustrate the satisfactory performance of the ILS localizer maneuvers.

Figure 78 illustrates the performance of the DFDC algorithm for 90° captures at various ranges. The performance of the captures and track was satisfactory. Captures of 7, 6, 5, 4, and 3 NM had the initial heading changed from 90° to 80°, 70°, 65°, and 45°, respectively, so that a reasonable overshoot would be maintained.

Figure 79 illustrates the effect of a 30 knot cross wind with beam noise, these simulation results demonstrate the satisfactory performance of the DFDC capture algorithm with a cross wind. Analog track wind has little effect on the capture and track performance.

Figure 80 illustrates the effects of a 30 knot cross wind with beam noise and gust. These simulation results demonstrate the satisfactory performance of the DFDC capture and track algorithm.

5.4.1.2 Propeller Powered Aircraft - 120 knots

Figures 81 through 85 illustrate the performance of the DFDC ILS algorithm for heading hold, capture, and track modes for the ILS localizer with a propeller powered aircraft flying at 120 knots for various heading intercept angles and iterations of distance (ATD) of 5, 10, and 15 NM respectively, from threshold. These hybrid simulation results illustrate the satisfactory performance of the ILS localizer maneuvers.

Figure 84 illustrates the effect of a 30 knot cross wind with beam noise. These simulation results demonstrate the satisfactory performance of the DFDC capture algorithm.

Figure 85 illustrates the effects of a 30 knot cross wind with beam noise and gust. These simulation results demonstrate the satisfactory performance of the DFDC capture and track algorithm.

5.4.1.3 Jet Aircraft - 150 knots

Figures 86 through 92 illustrate the performance of the DFDC ILS algorithm for heading hold, capture, and track modes for the ILS localizer with a jet aircraft flying at 150 knots for various heading intercept angles and iterations of distance (ATD of 5, 10, 15, and 20 NM respectively, from threshold. These hybrid simulation results illustrate the satisfactory performance of the ILS localizer maneuvers.

Figure 91 illustrates the effect of a 30 knot cross wind with beam noise. These simulation results demonstrate the satisfactory performance of the DFDC capture algorithm.

Figure 92 illustrates the effects of a 30 knot cross wind with beam noise and gust. These simulation results demonstrate the satisfactory performance of the DFDC capture and track algorithm.

The effects of airspeed on the capture maneuver are illustrated in Figure 90 . The airspeed was varied from 120 knots to 240 knots in 10 knot steps. These simulation results demonstrate the adaptability of the circular capture algorithm to airspeed perturbations.

5.4.2 ILS Glideslope

Figures 93, 94, and 95 illustrate the performance of the DFDC ILS glideslope algorithm for glideslope capture and track for the jet transport, the propeller powered aircraft, and the jet aircraft, respectively. The hybrid simulation results demonstrate the satisfactory performance of the glideslope capture and track modes with and without beam noise and wind as follows:

- 1) Part a of the figures demonstrate that for no disturbances the aircraft has commanded to capture and track the glideslope beam within 10 ft. At threshold, the radio deviation had a typical value of 25uA.
- 2) Parts b and c of the Figures demonstrate that a 30 knot along track wind had little effect on the performance as compared to the no wind case (part a).
- 3) Parts d, e, f, g, and h of the Figures demonstrate the satisfactory performance of the DFDC for increasing glideslope beam noise levels.

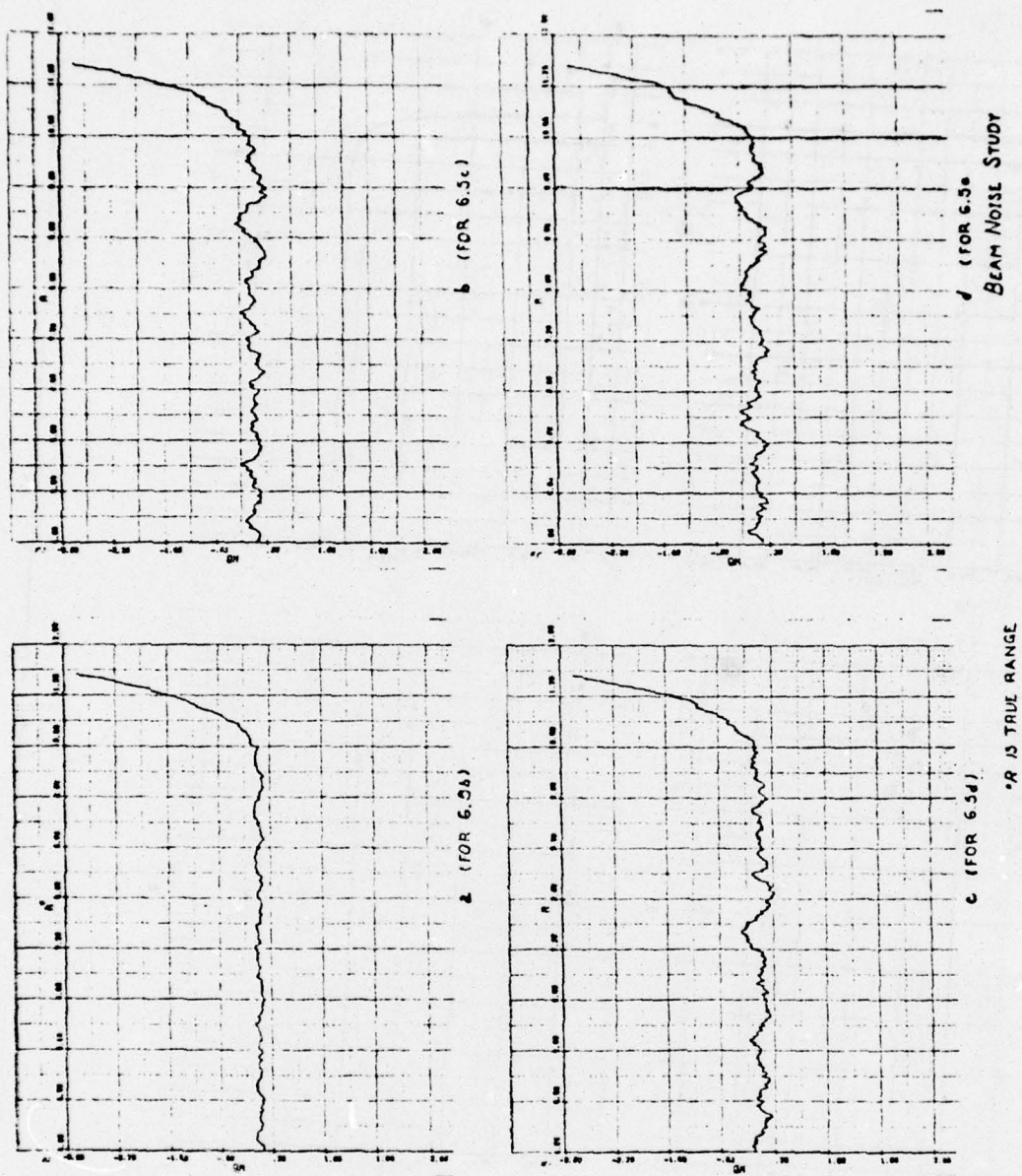


Figure 57. Beam Noise Study - Sampled Beams

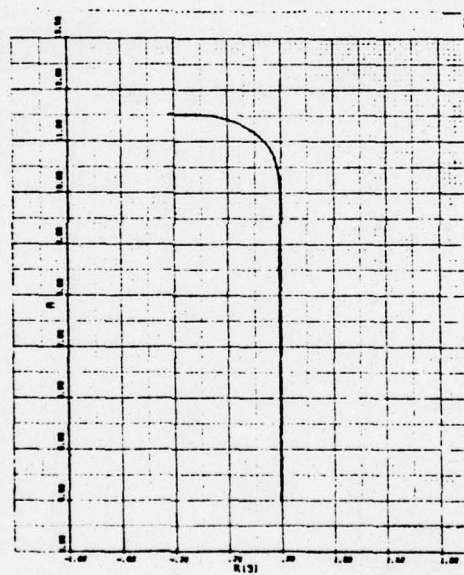
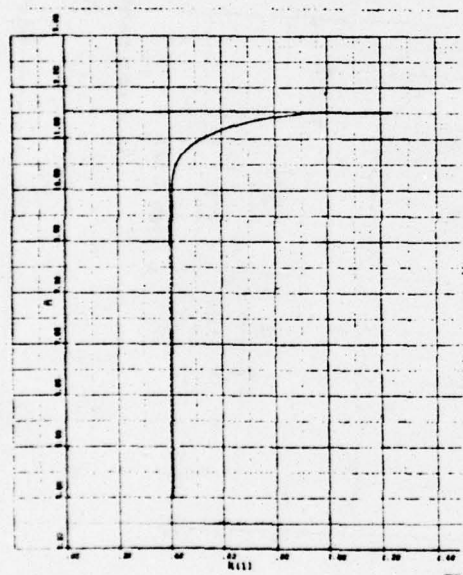
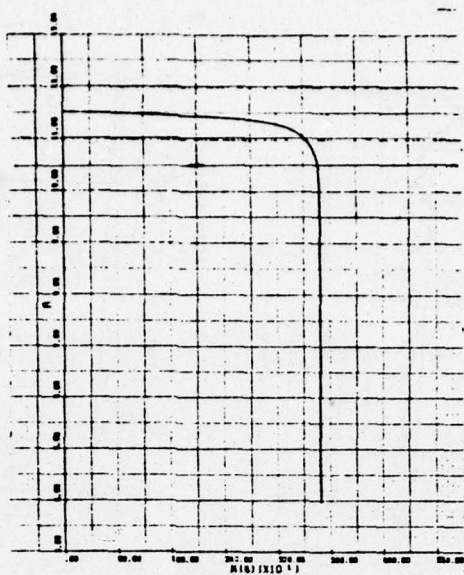
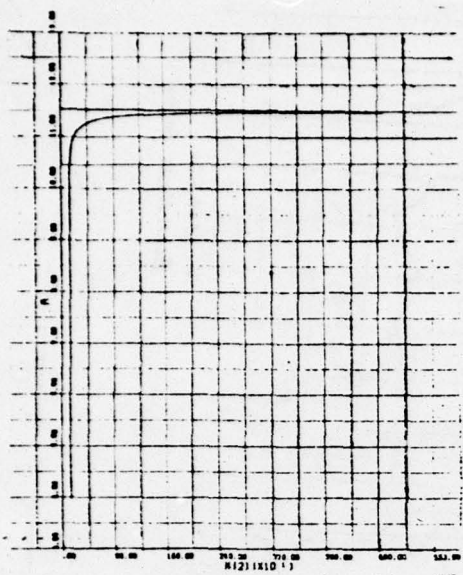


Figure 58a. Beam Noise Study - Kalman Gains - For 6.5a

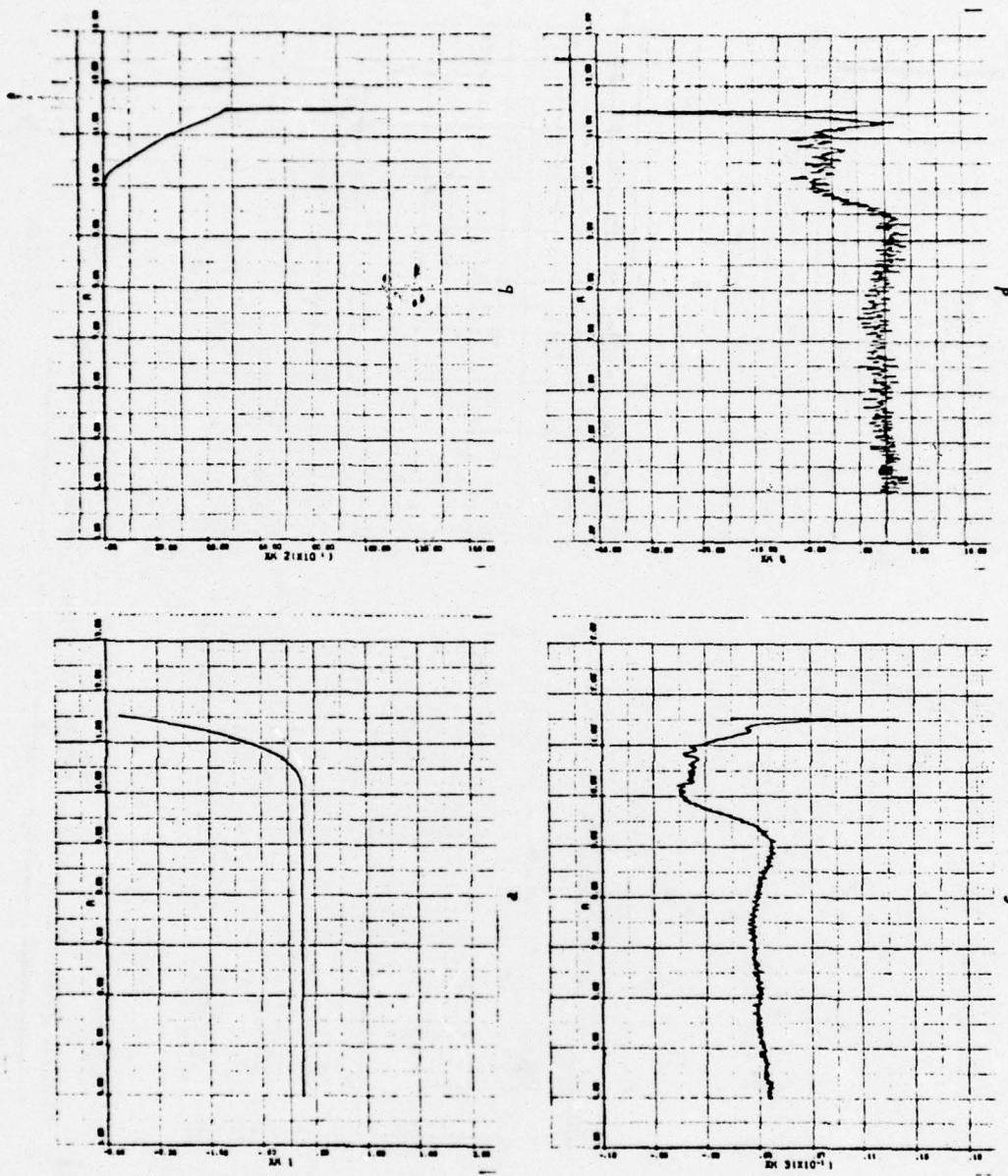


Figure 58b. Beam Noise Study - State Vector - For 6.5a

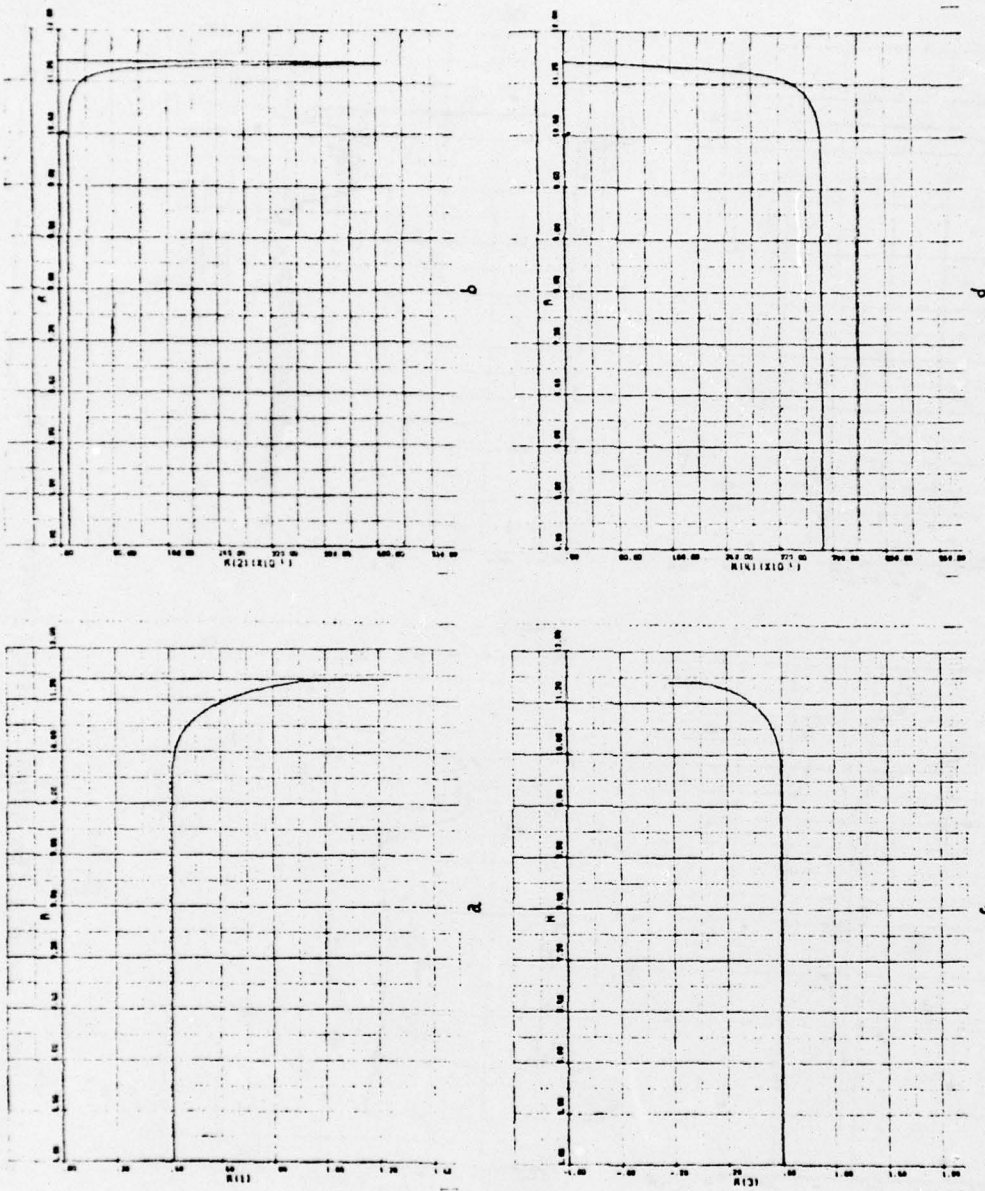


Figure 59a. Beam Noise Study - Kalman Gains - For 6.5b

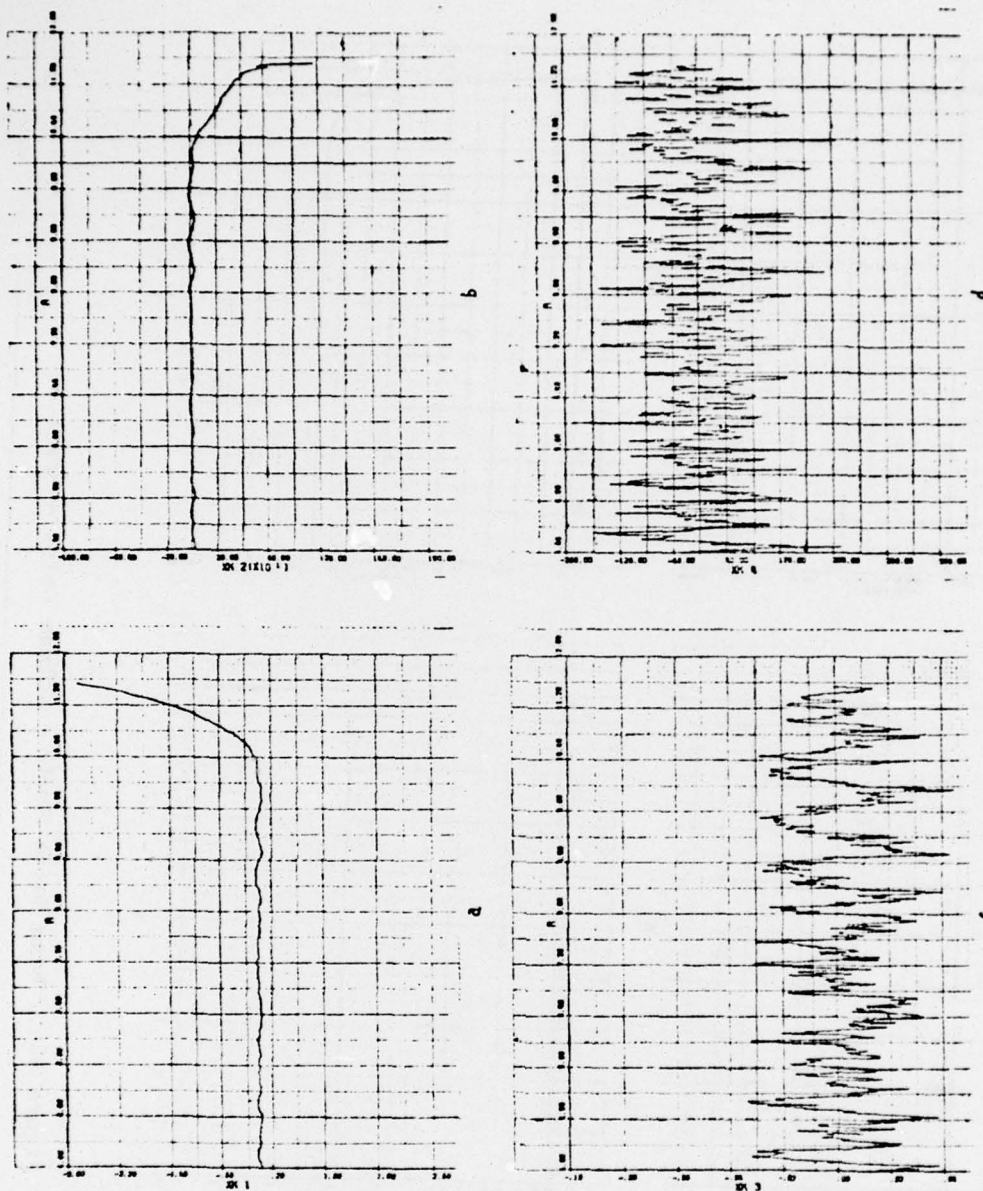


Figure 59b. Beam Noise Study - State Vector - For 6.5b

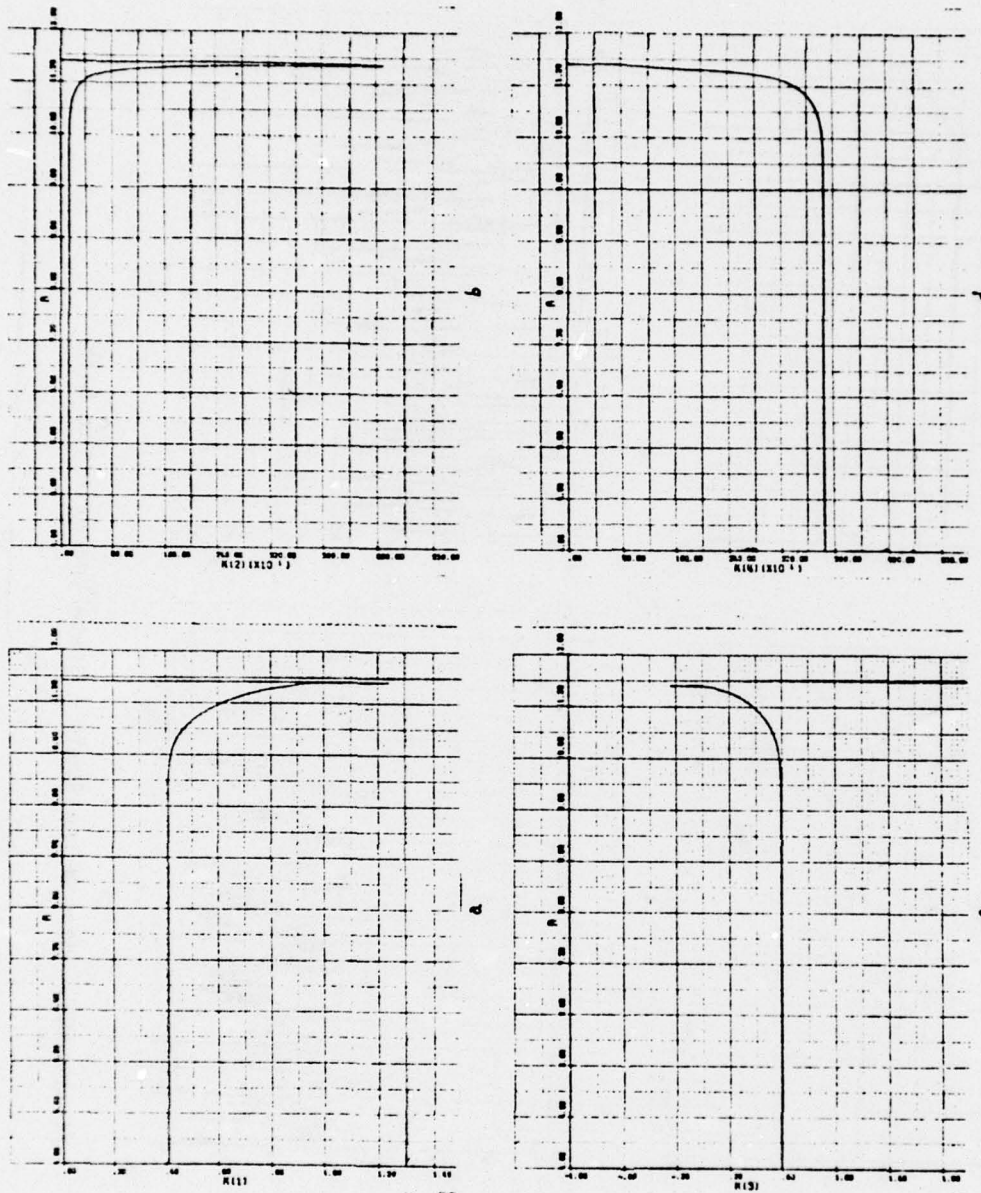


Figure 60a. Beam Noise Study - Kalman Gains - For 6.5c

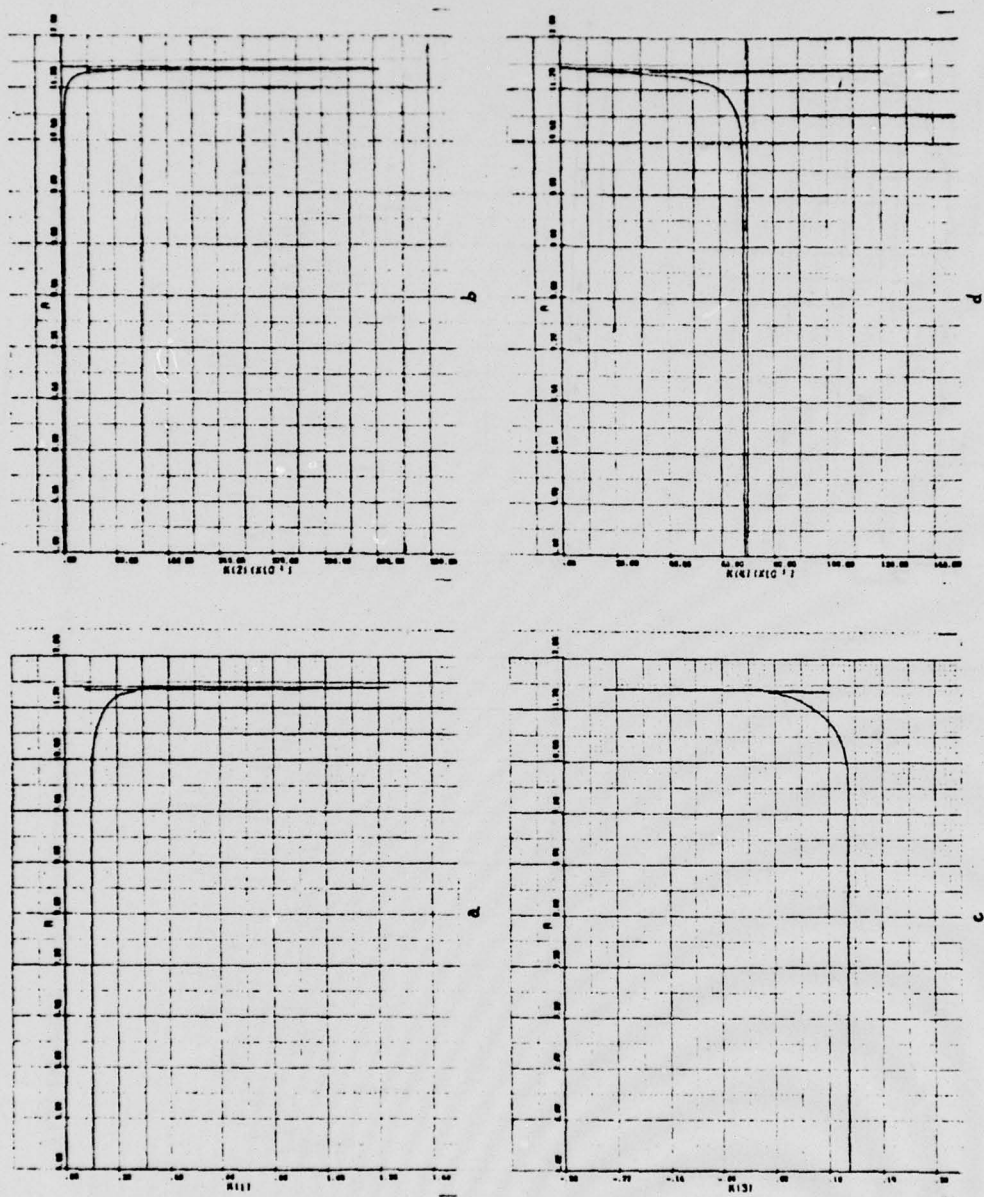


Figure 61a. Beam Noise Study - Kalman Gains - For 6.5d

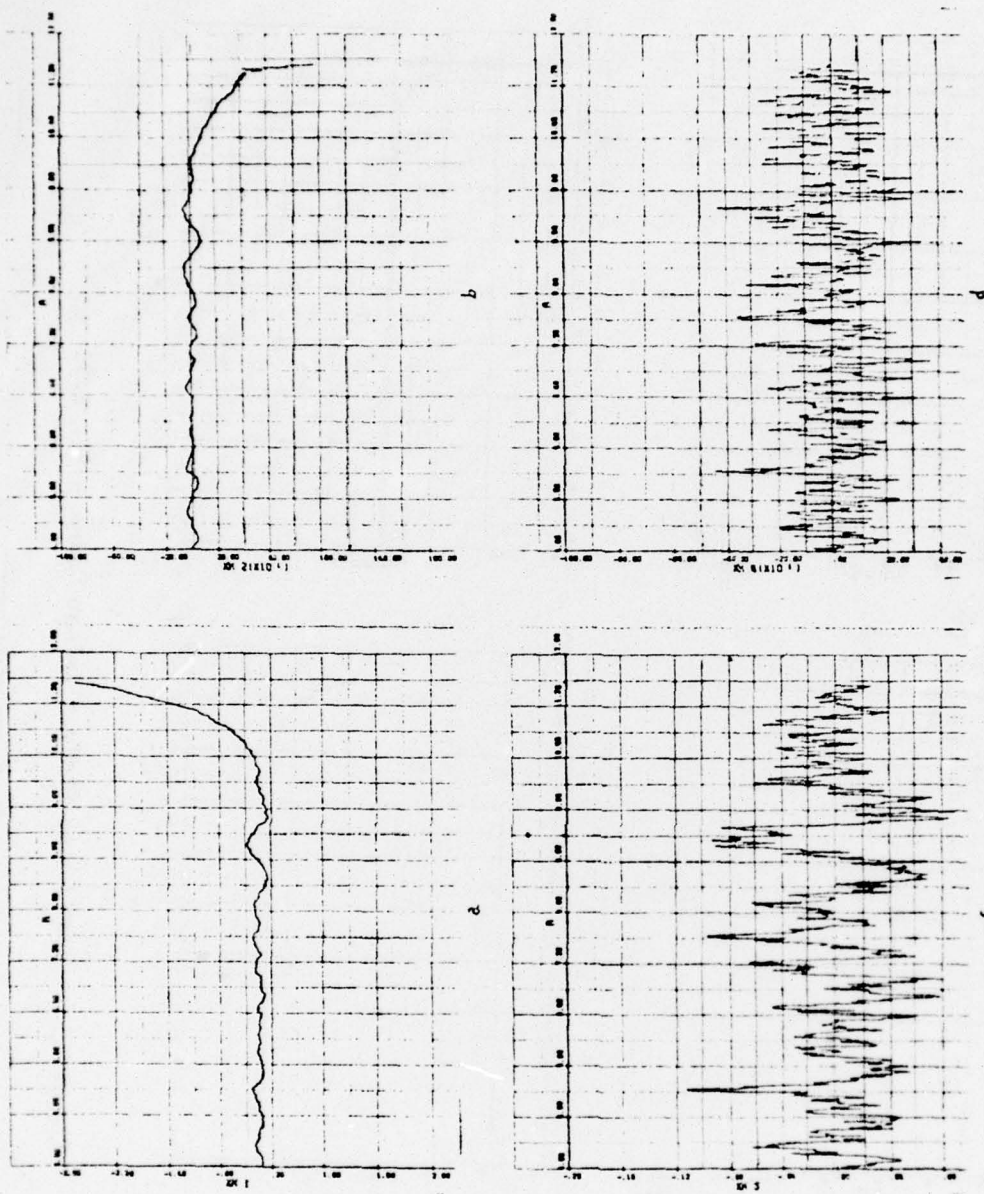


Figure 60b. Beam Noise Study - State Vector - For 6.5c

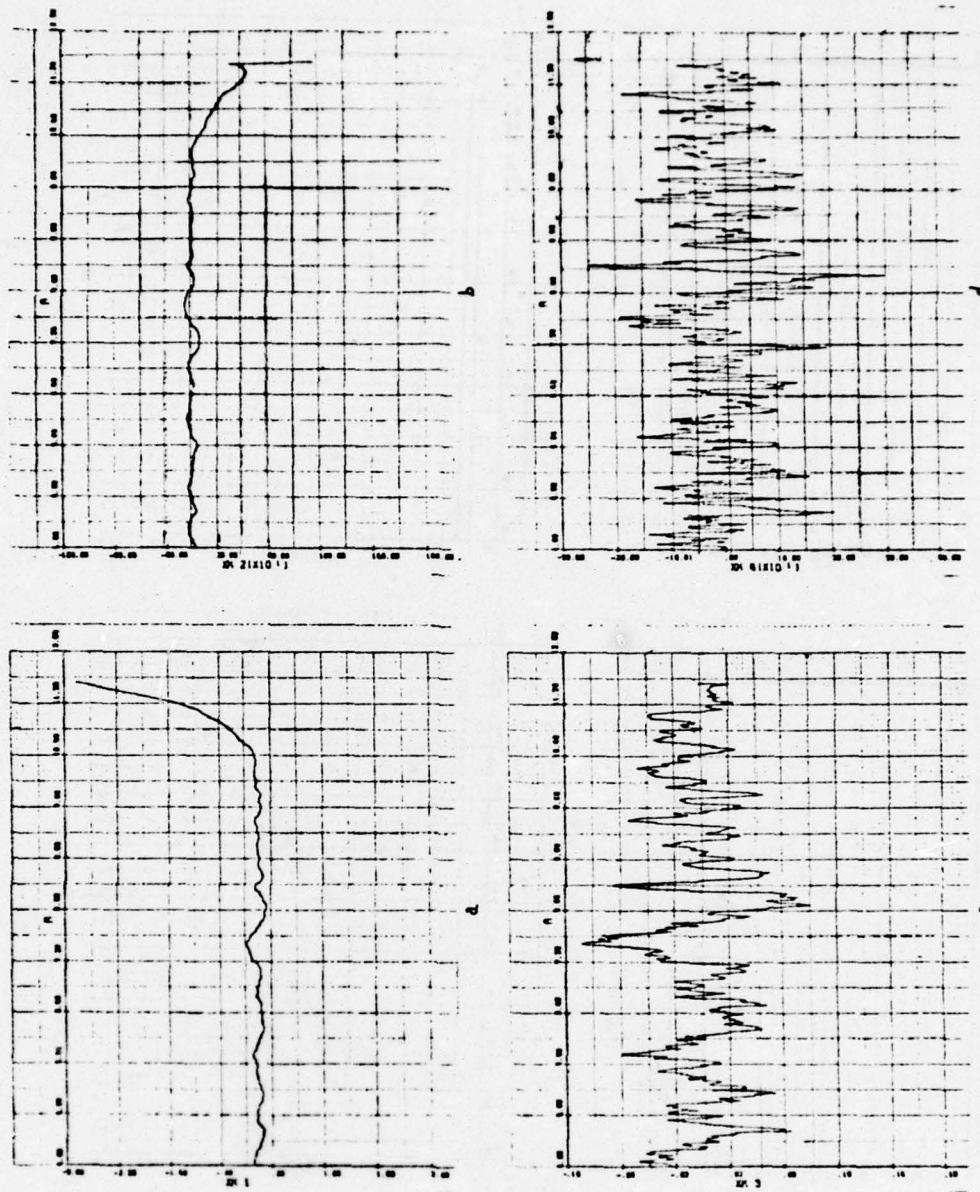


Figure 6lb. Beam Noise Study - State Vector - For 6.5d

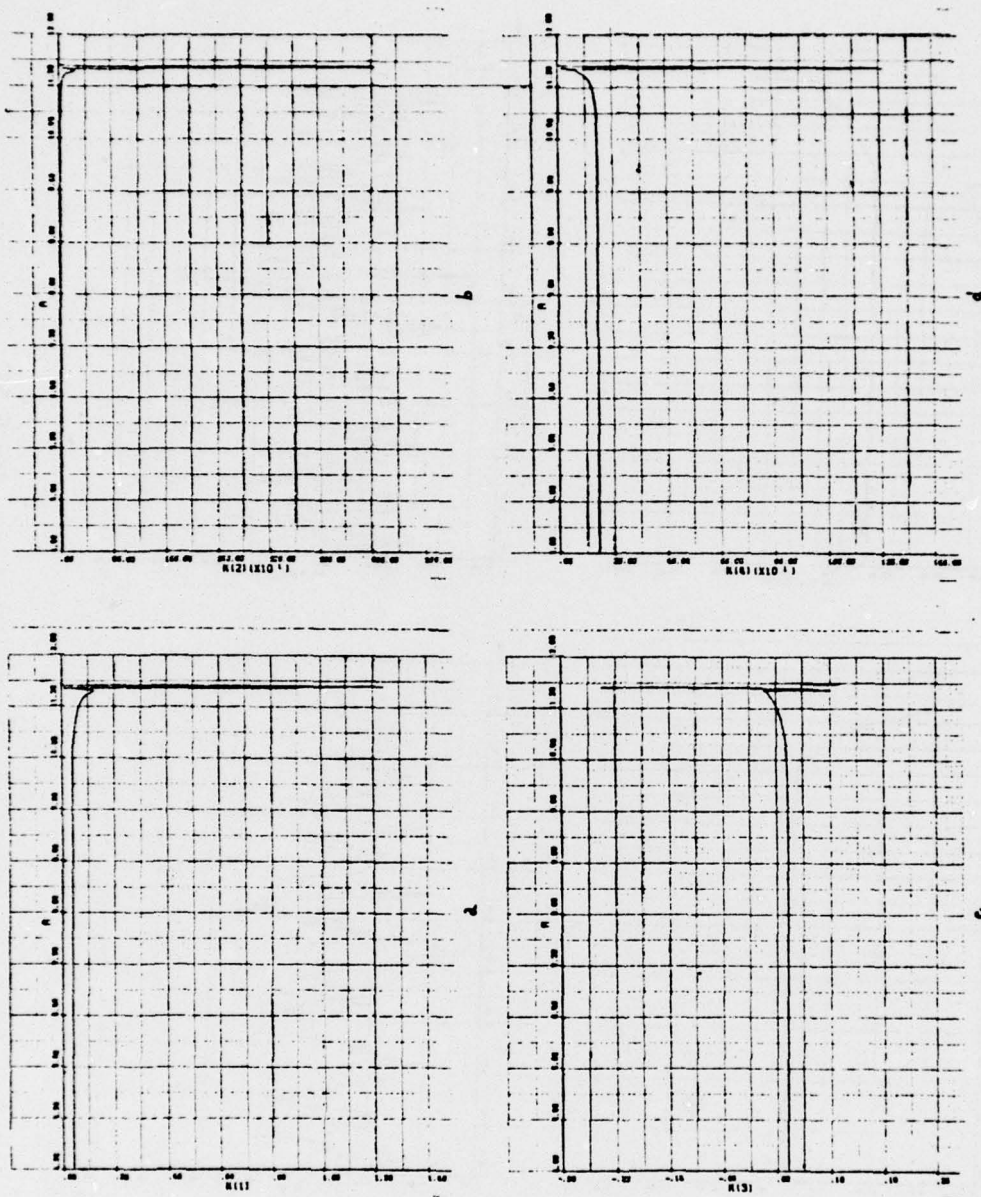


Figure 62a. Beam Noise Study - Kalman Gains - For 6.5e

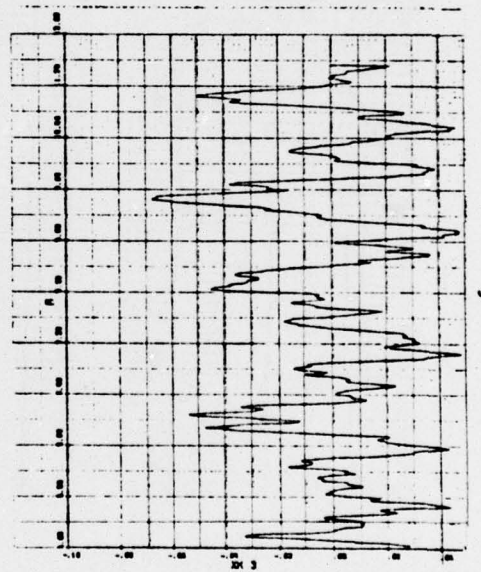
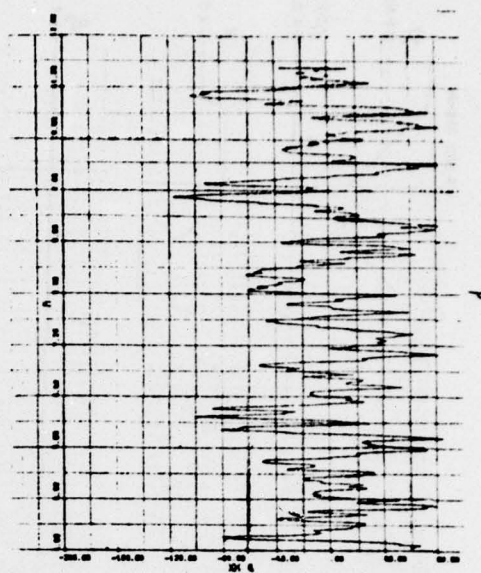
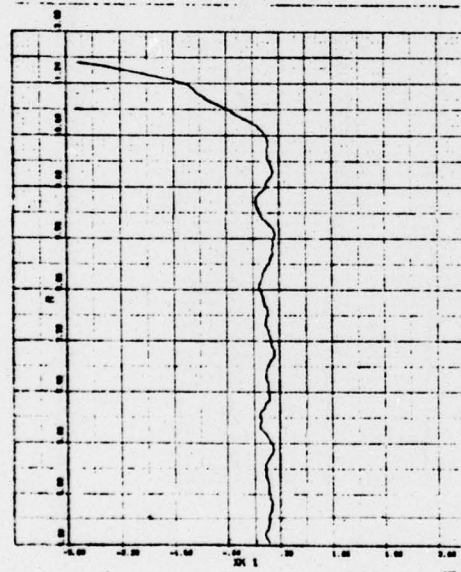
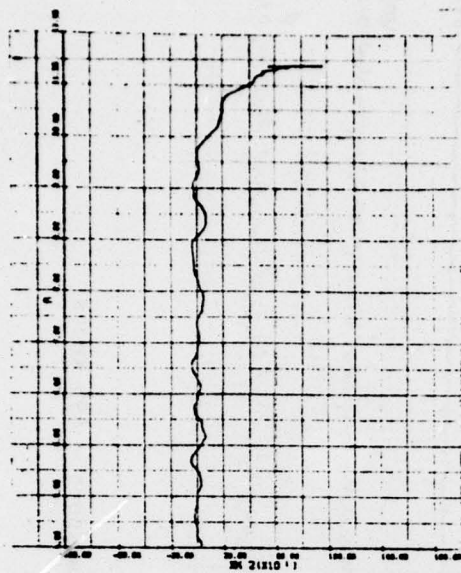


Figure 62b. Beam Noise Study - State Vector - For 6.5e

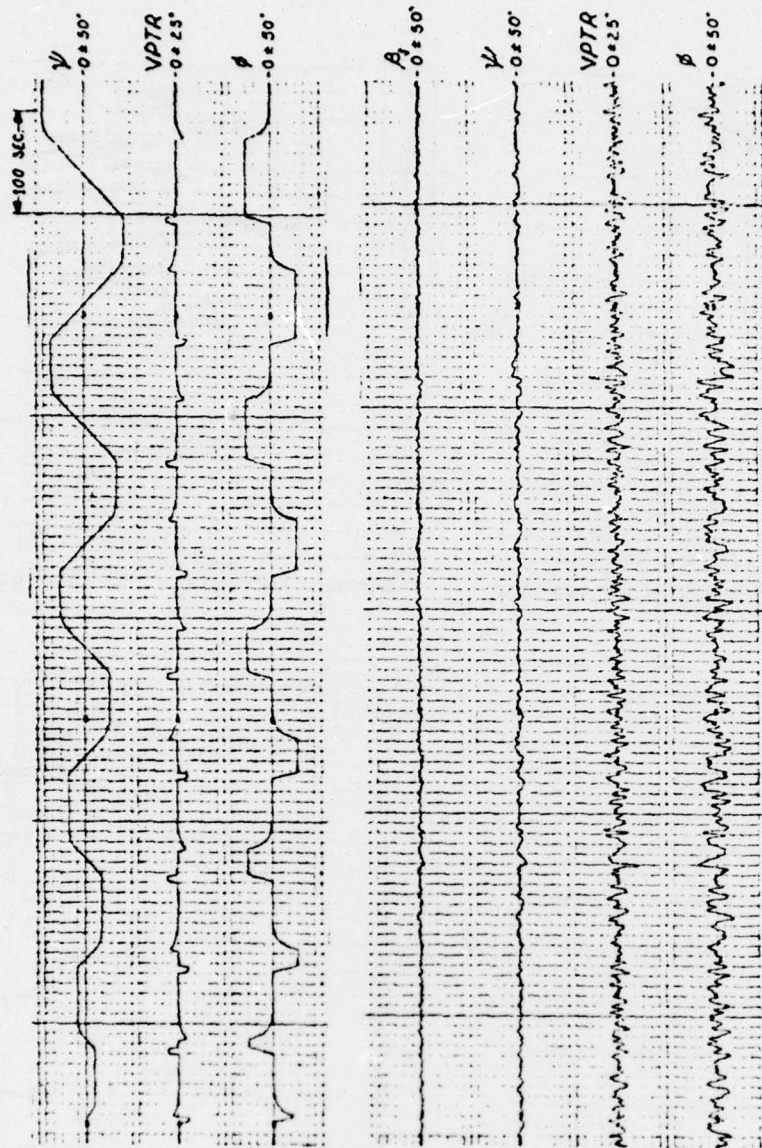


Figure 63. Manual Heading Mode - Jet Transport - 495 Knots

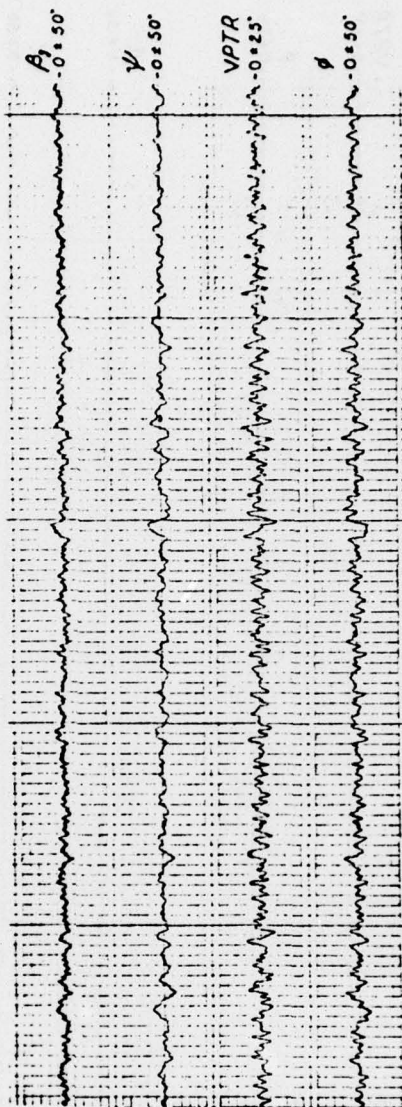
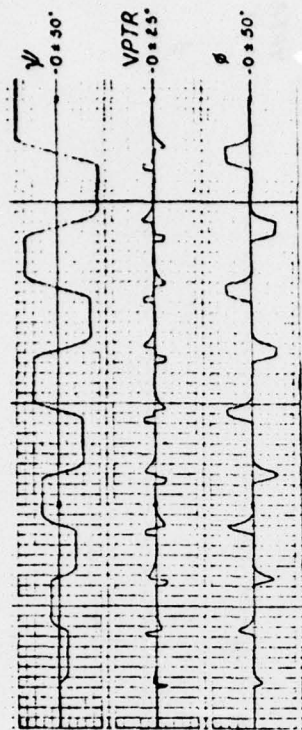


Figure 64. Manual Heading Mode - Jet Transport - 125 Knots

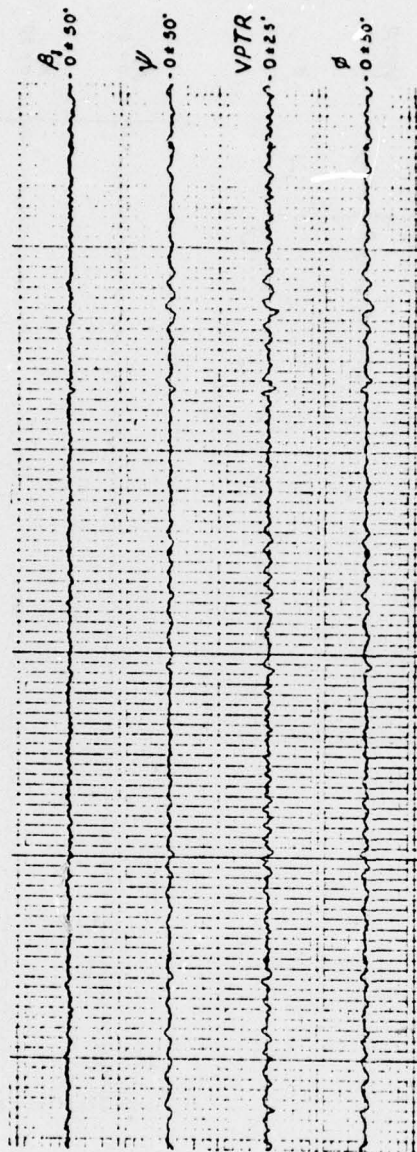
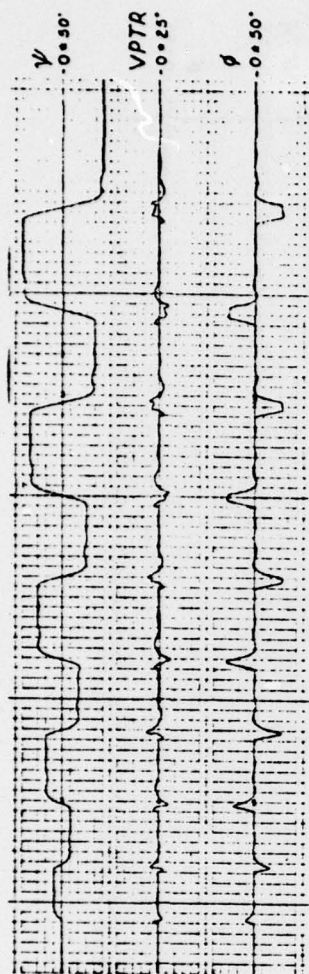


Figure 65. Manual Heading Mode - Propeller Powered A/C - 120 Knots

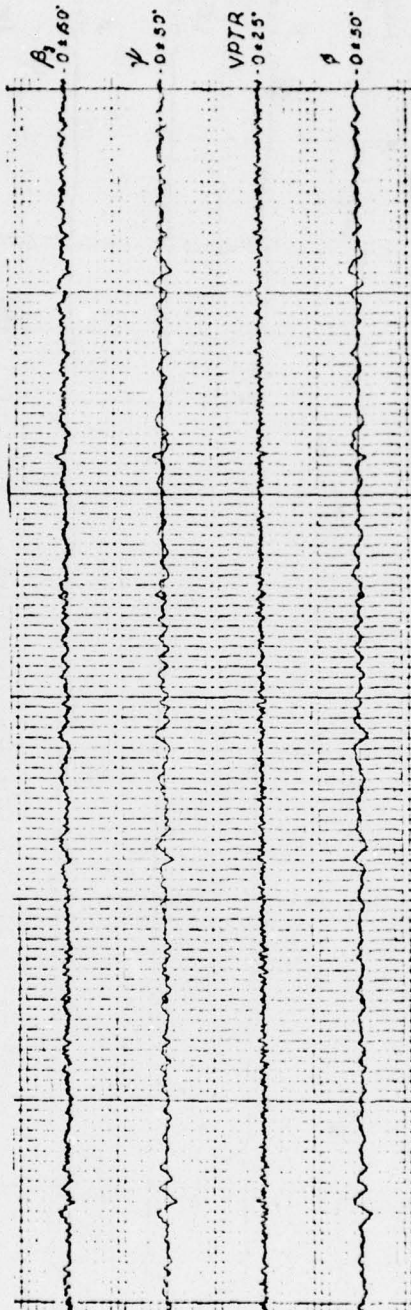
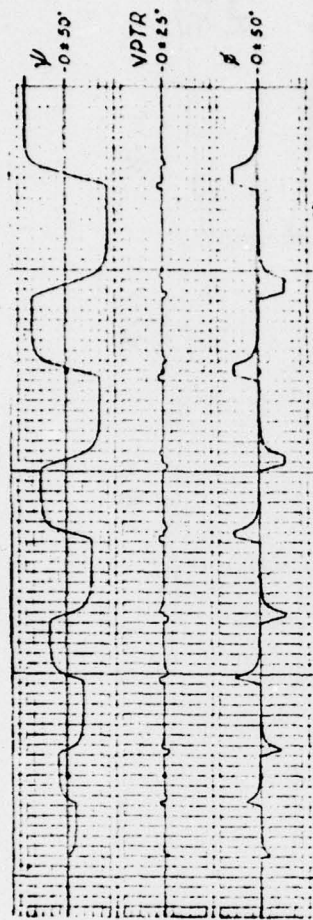


Figure 66. Manual Heading Mode - Jet Aircraft - 150 Knots

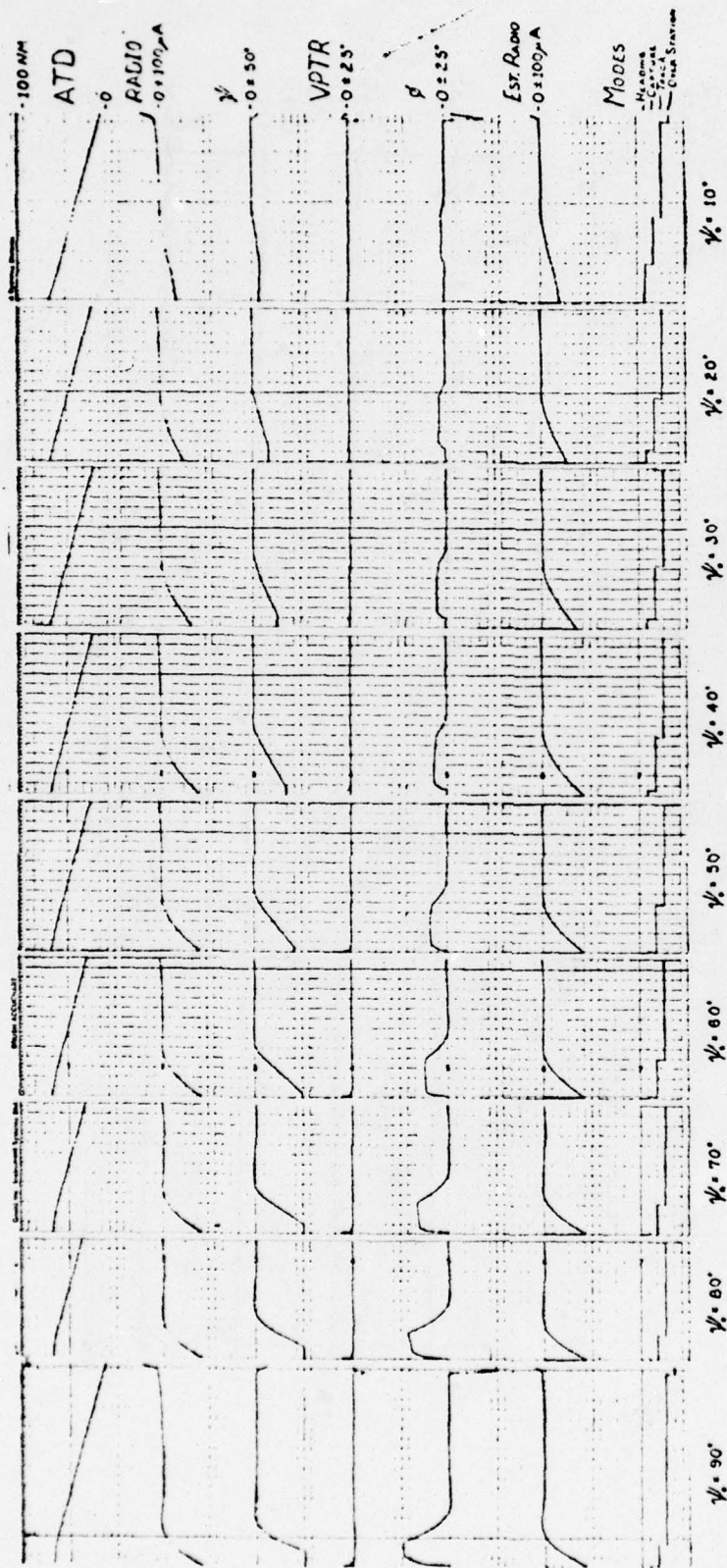


Figure 67. TACAN Mode, ATD = 70 NM, $\Delta\psi$

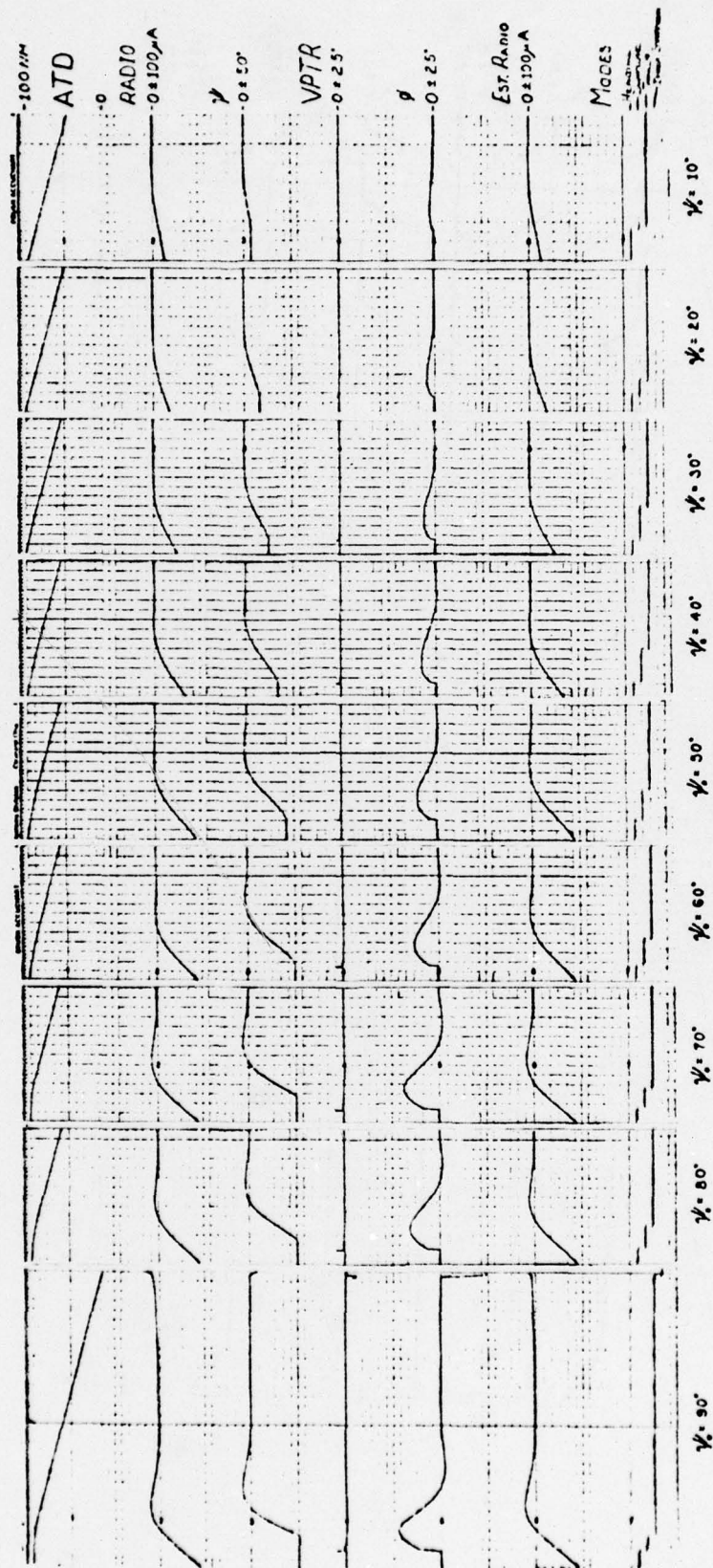


Figure 68. TACAN Mode, ATD = 100 NM, $\Delta\psi$

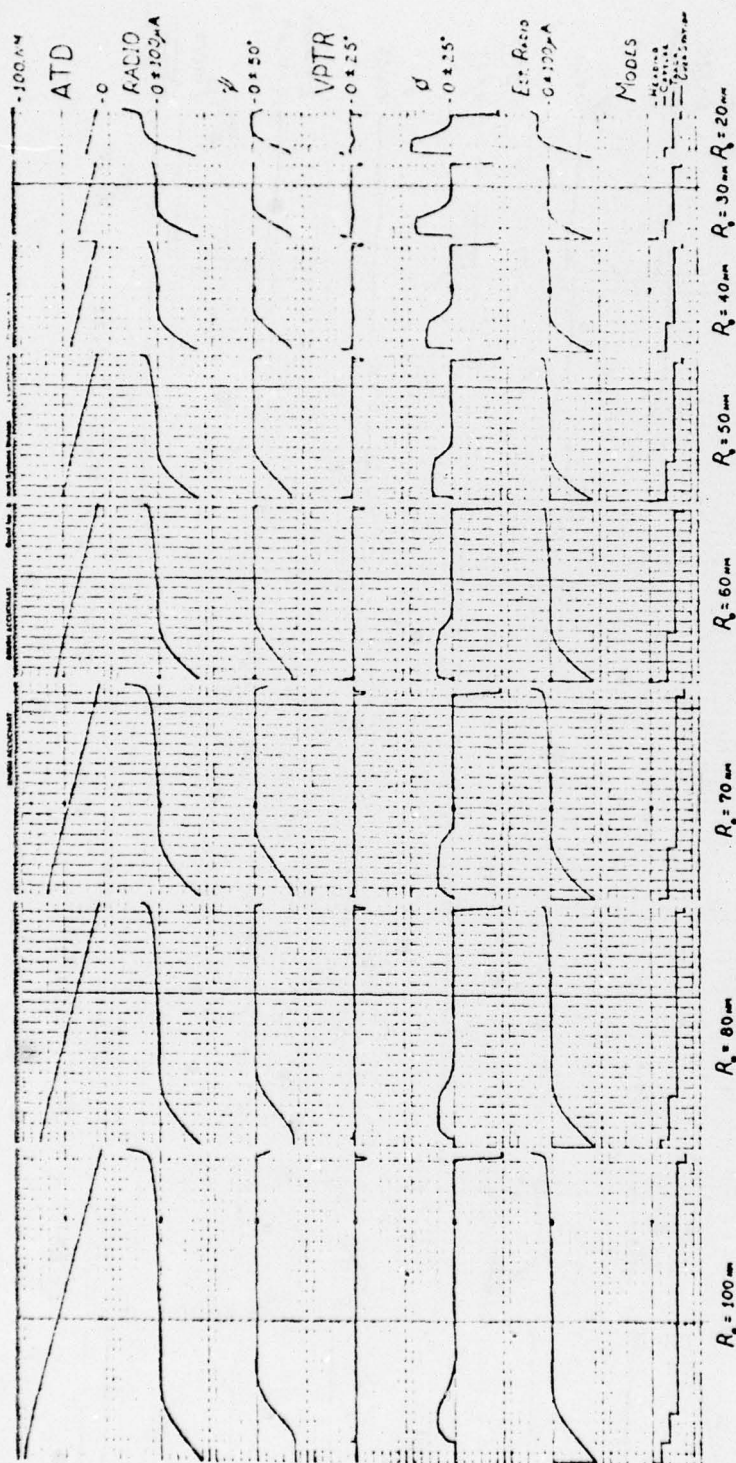


Figure 69a. TACAN Mode, Δ ATD, $\psi = 45^\circ$

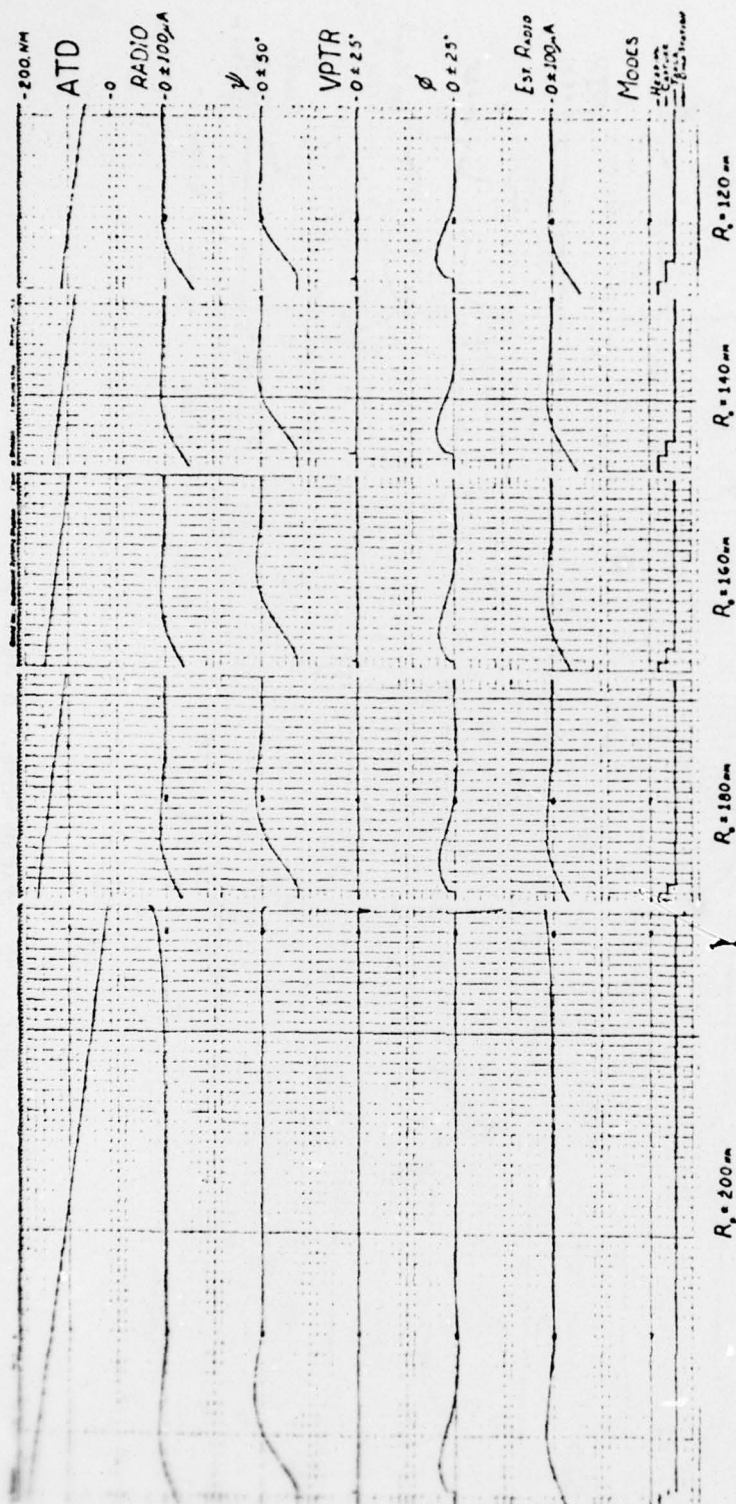


Figure 69b. TACAN Mode, Δ ATD, $\psi = 45^\circ$

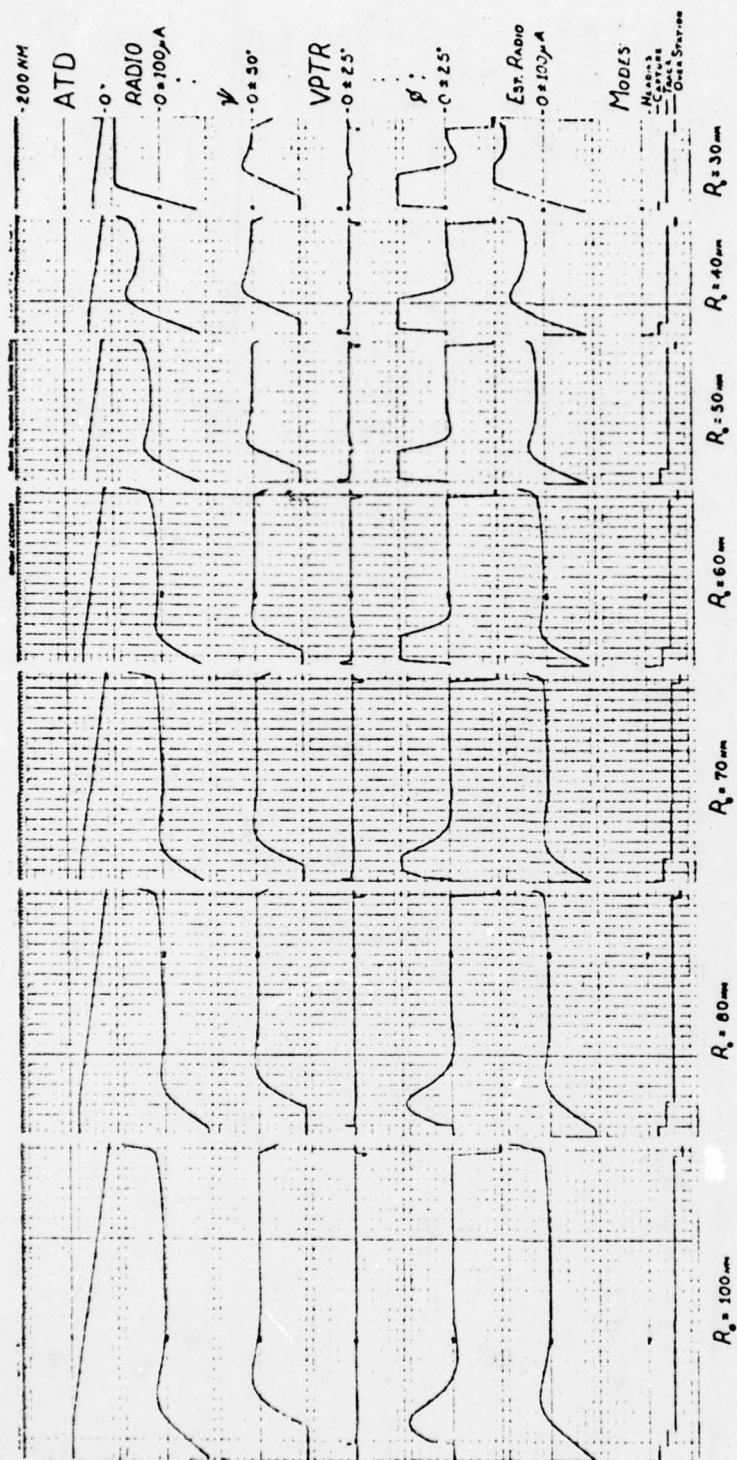


Figure 70. TACAN Mode, Δ ATD, $\psi = 90^\circ$

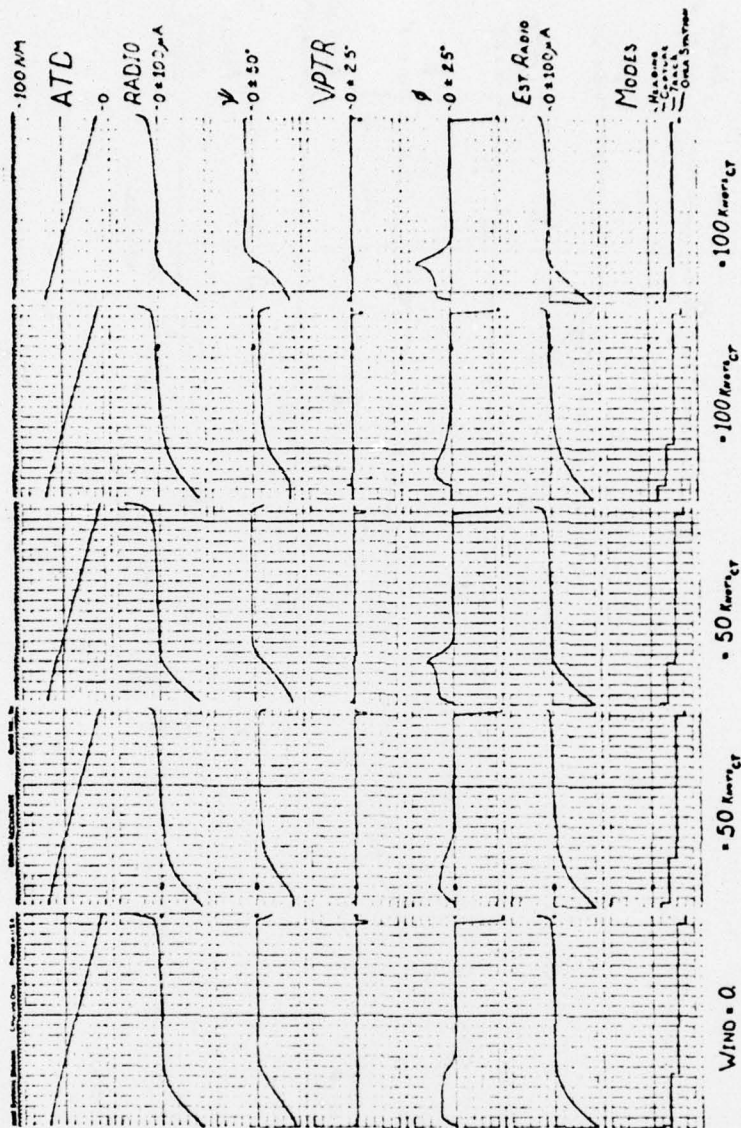


Figure 71. TACAN Mode, ATD = 70 NM, $\psi = 45^\circ$, Wind

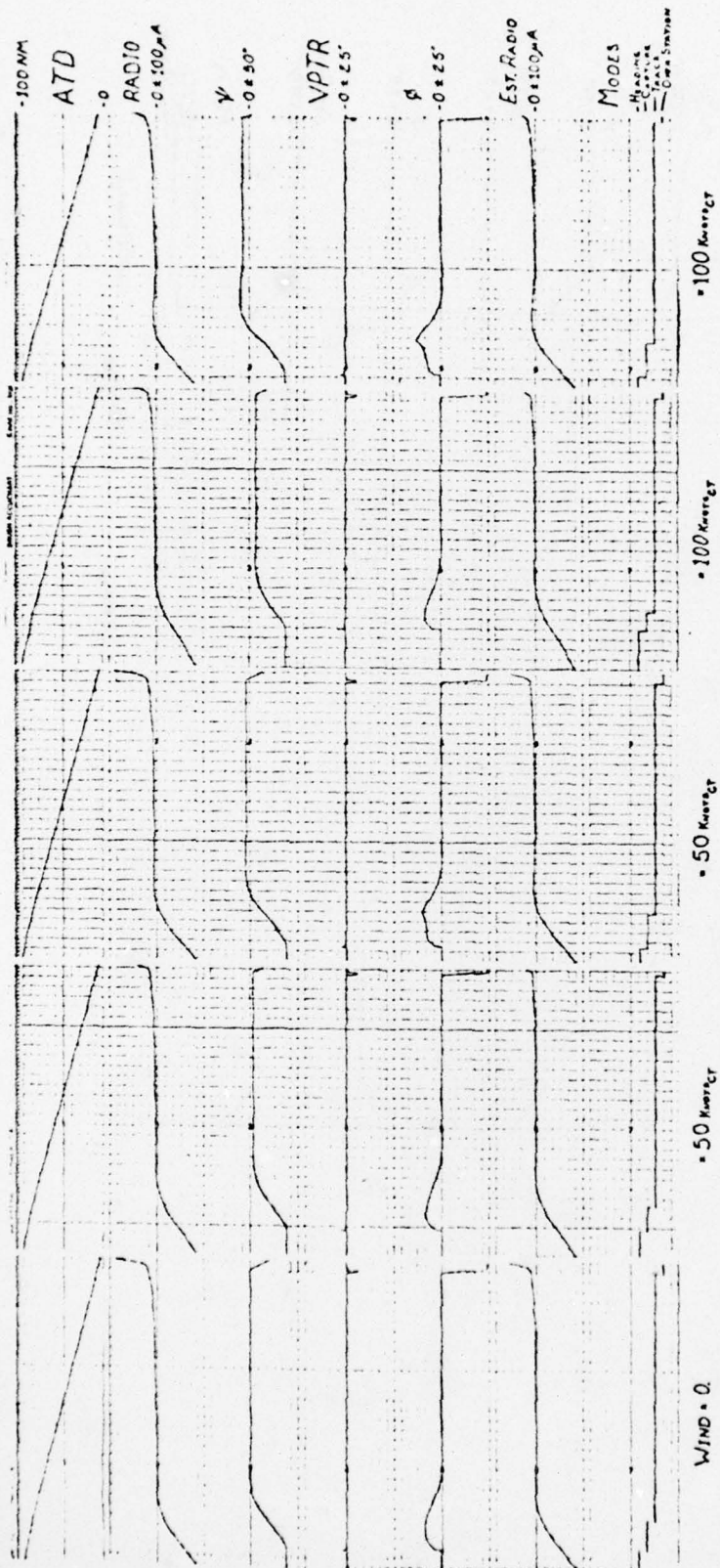


Figure 72. TACAN Mode, ATD = 100 NM, $\psi = 45^\circ$, Wind

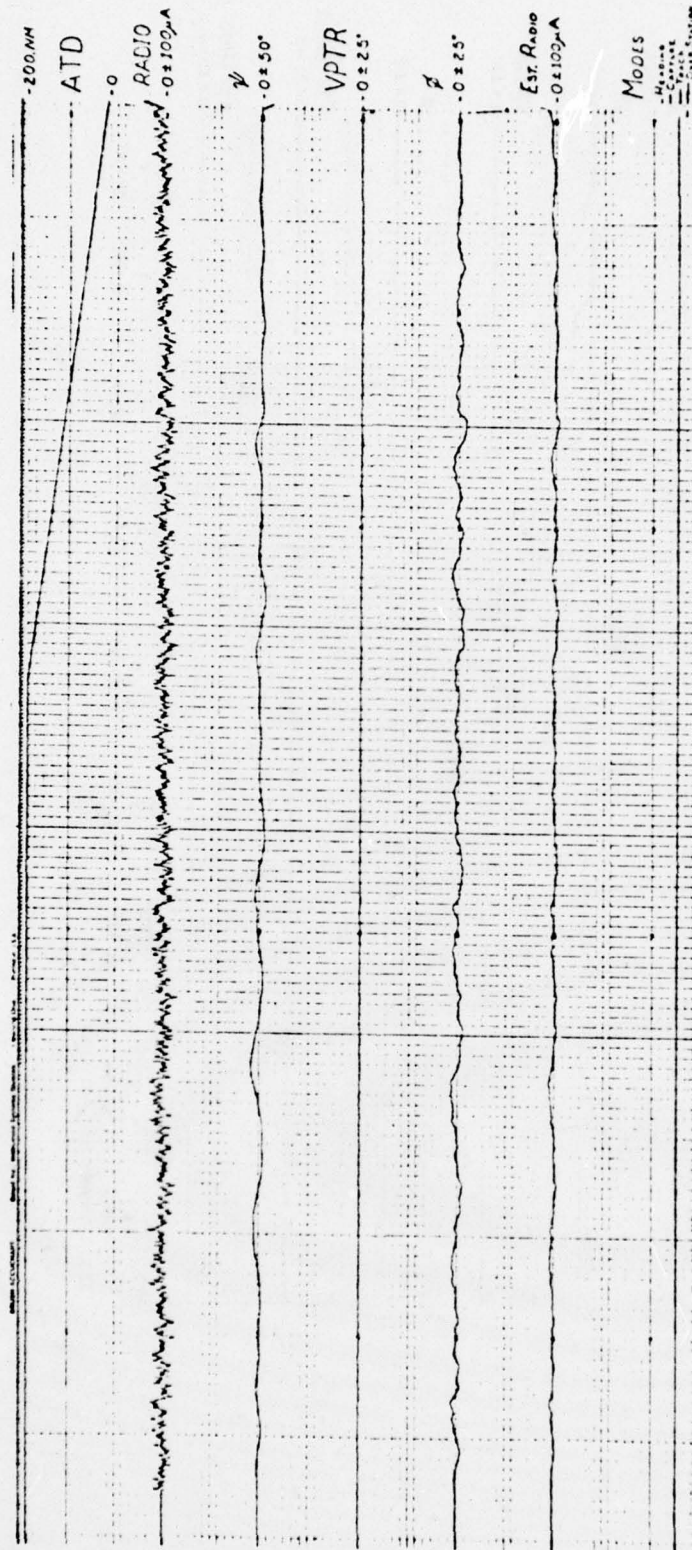


Figure 73. TACAN Mode, ATD = 200 NM, 10 μA_{rms} Radio Noise

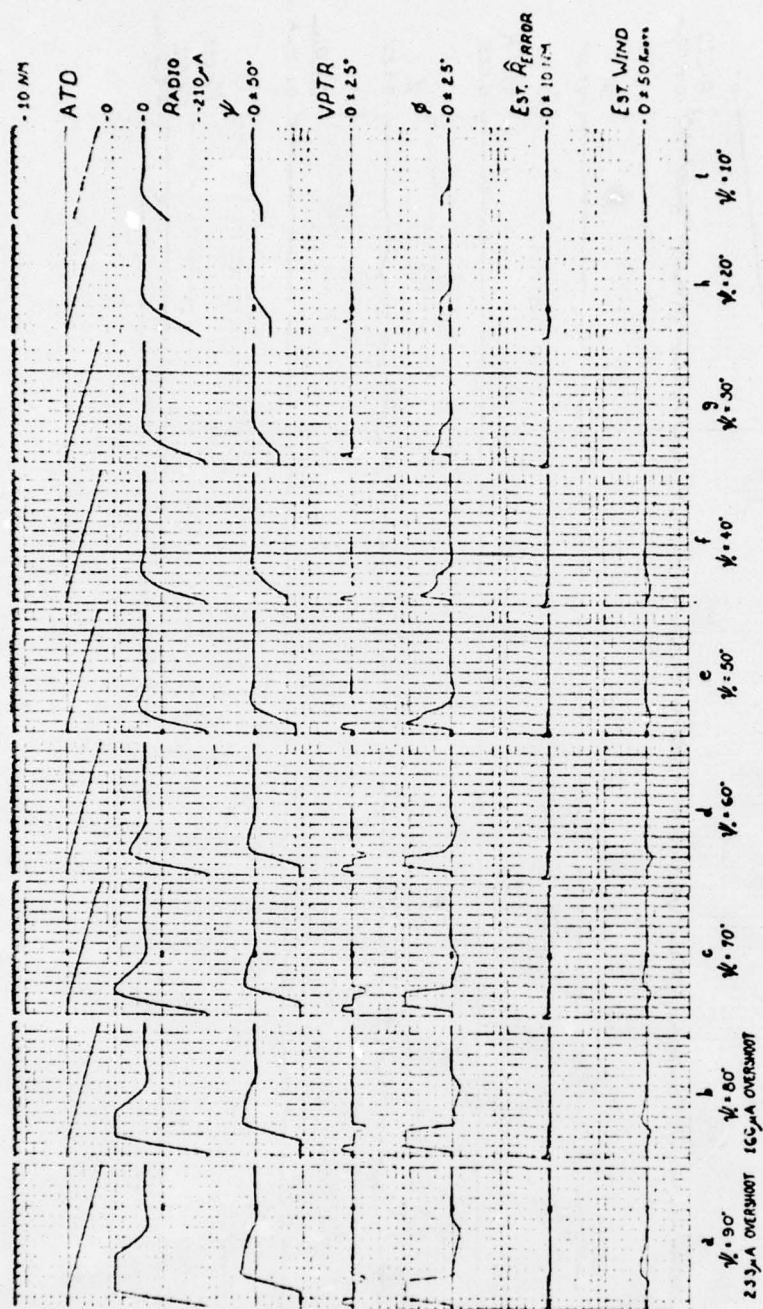


Figure 74. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 5 NM, $\Delta\psi$

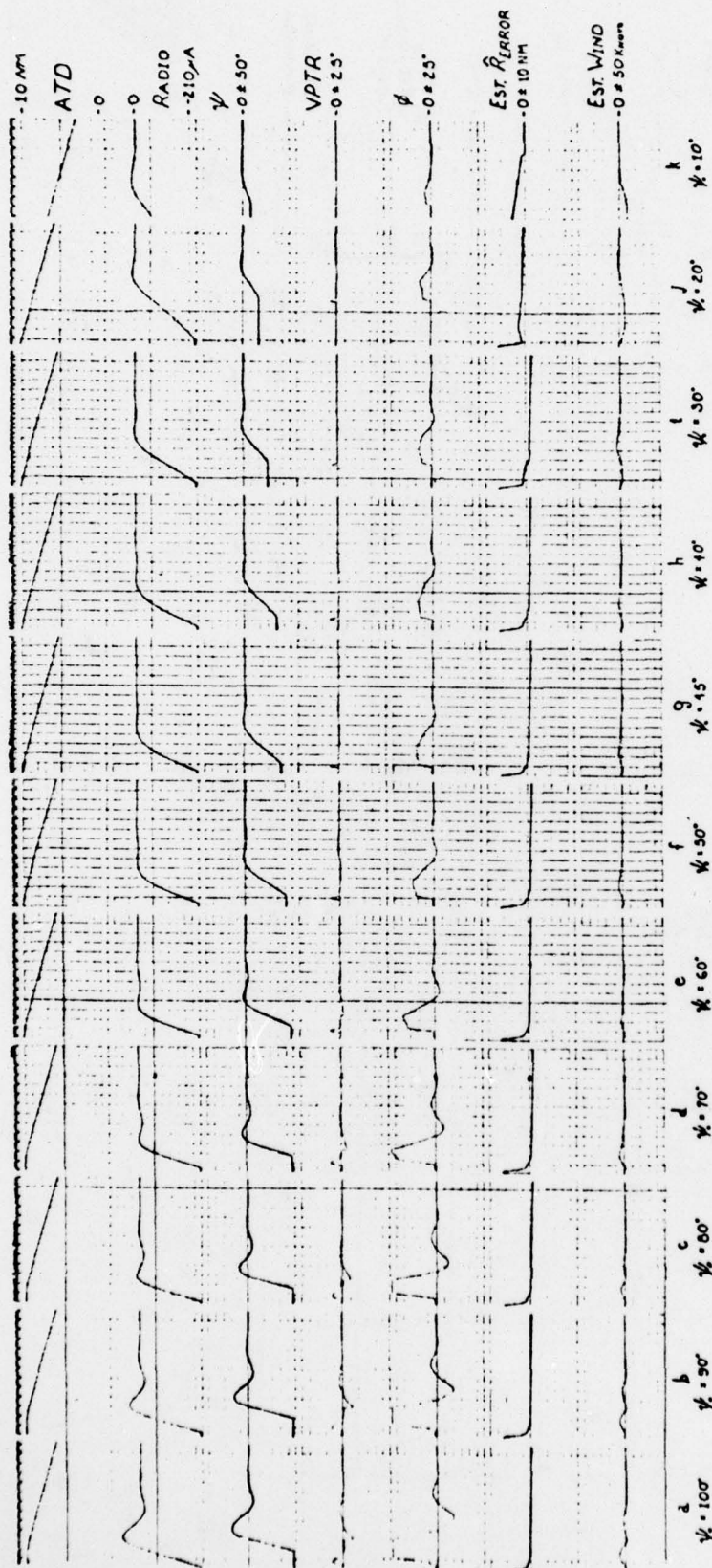


Figure 75. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 10 NM, $\Delta\psi$

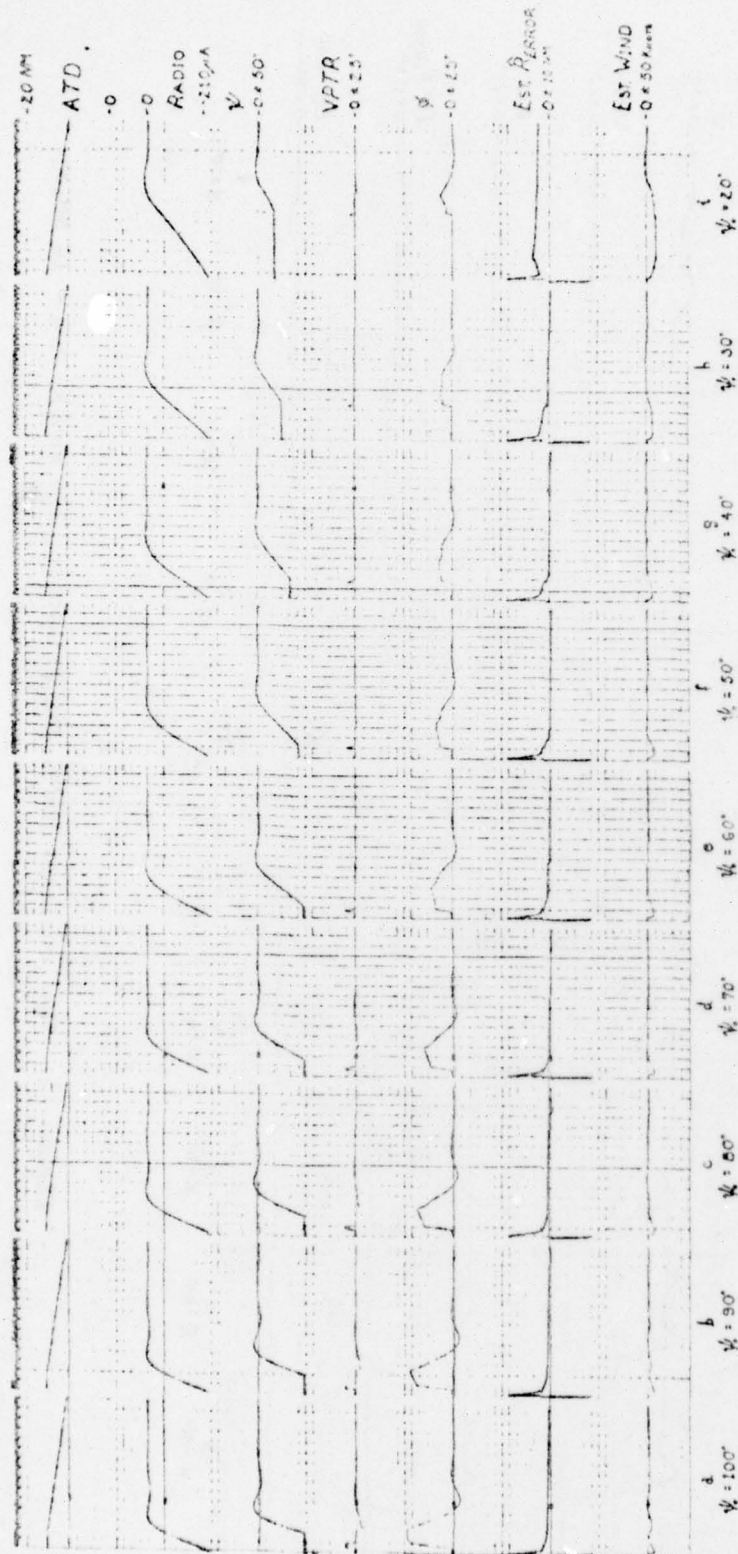


Figure 76. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 15 NM, 4°

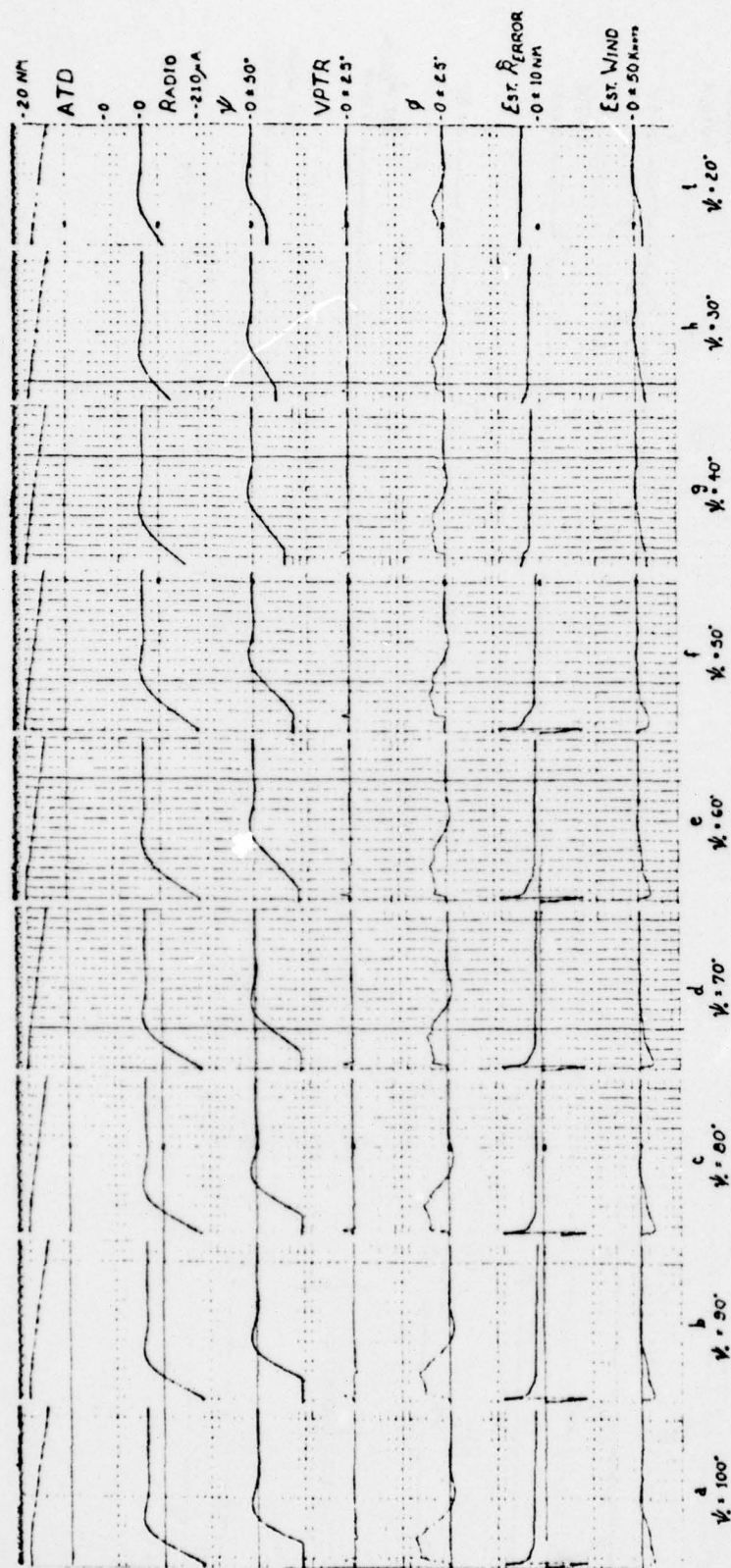


Figure 77. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 20 NM, $\Delta\psi$

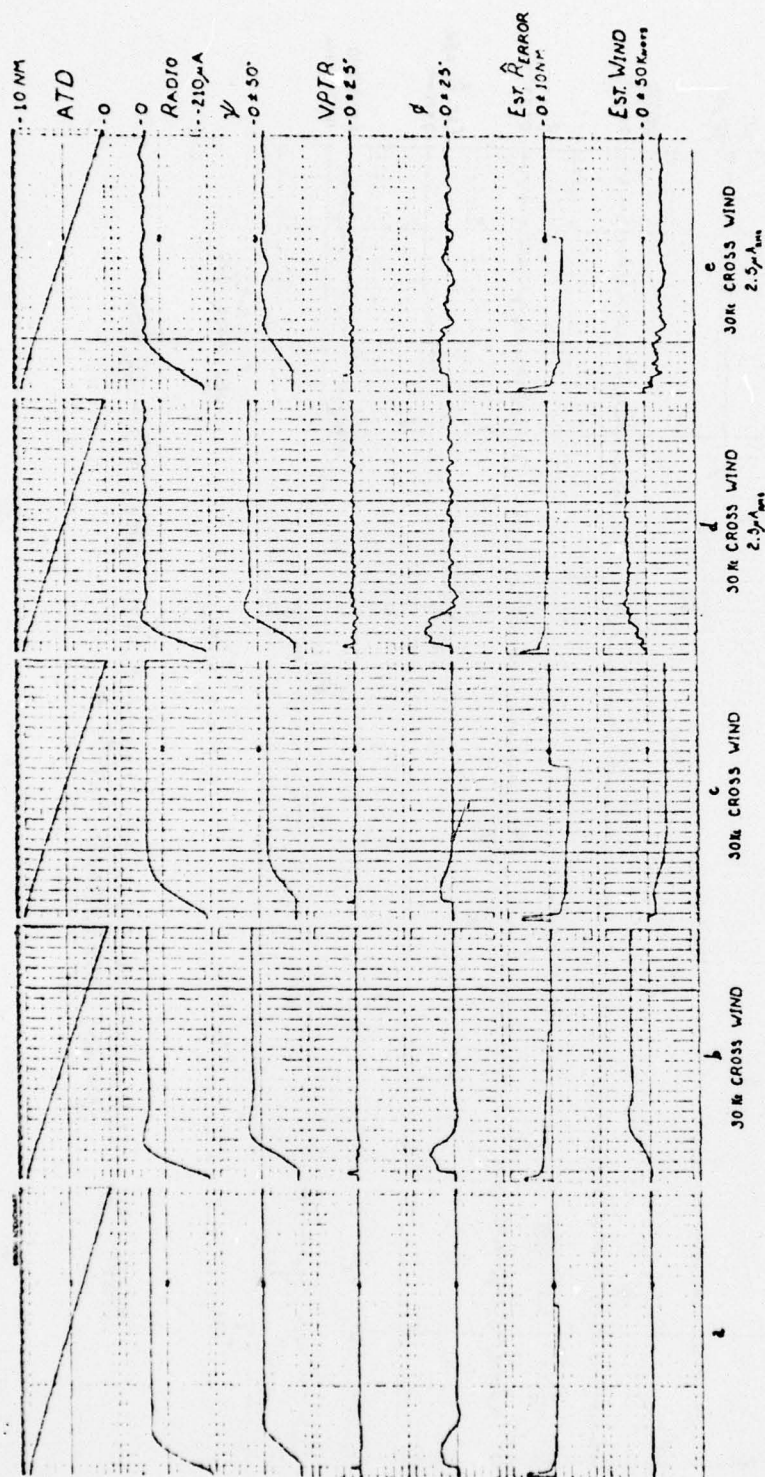


Figure 79. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 10 NM, $\gamma' = 45^\circ$, Wind, 2.5 μ Arms Radio Noise

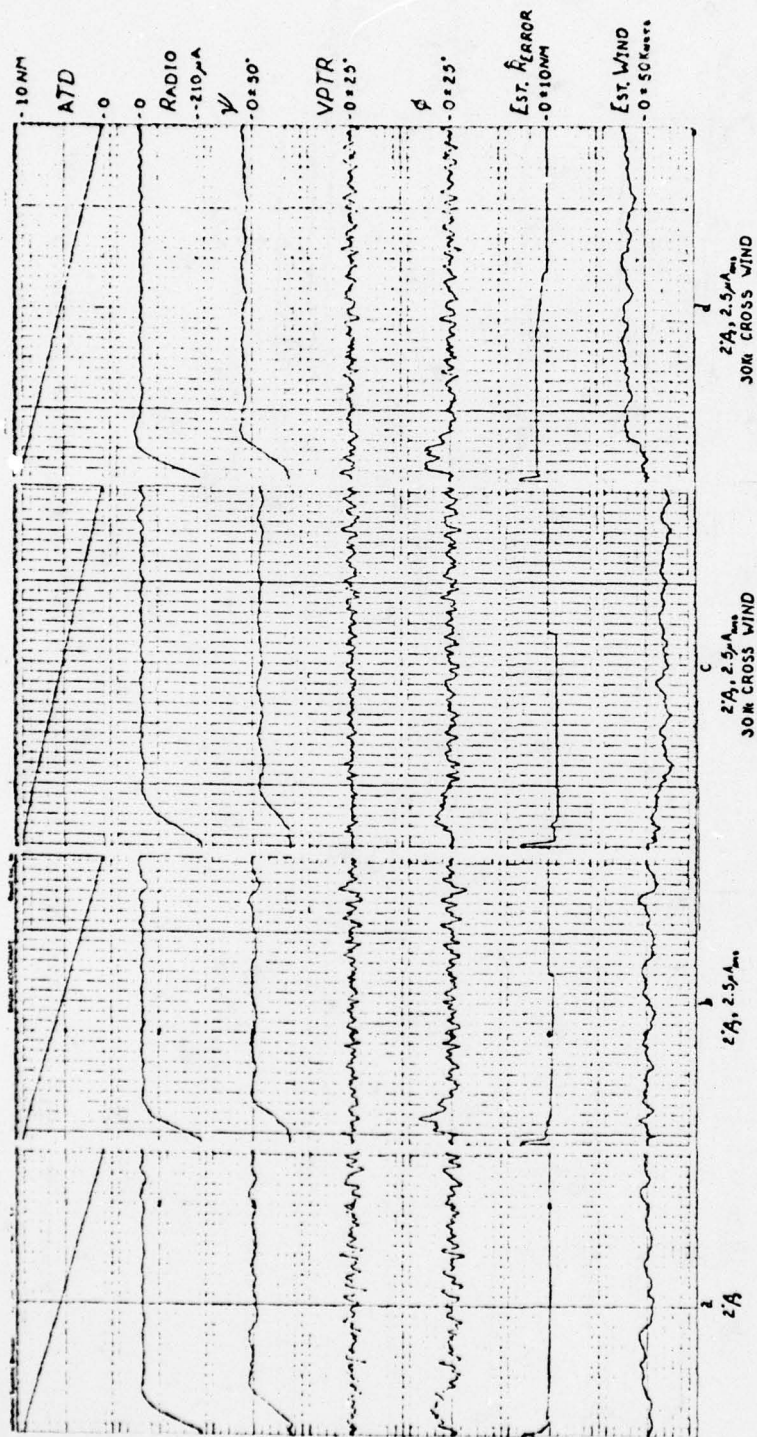
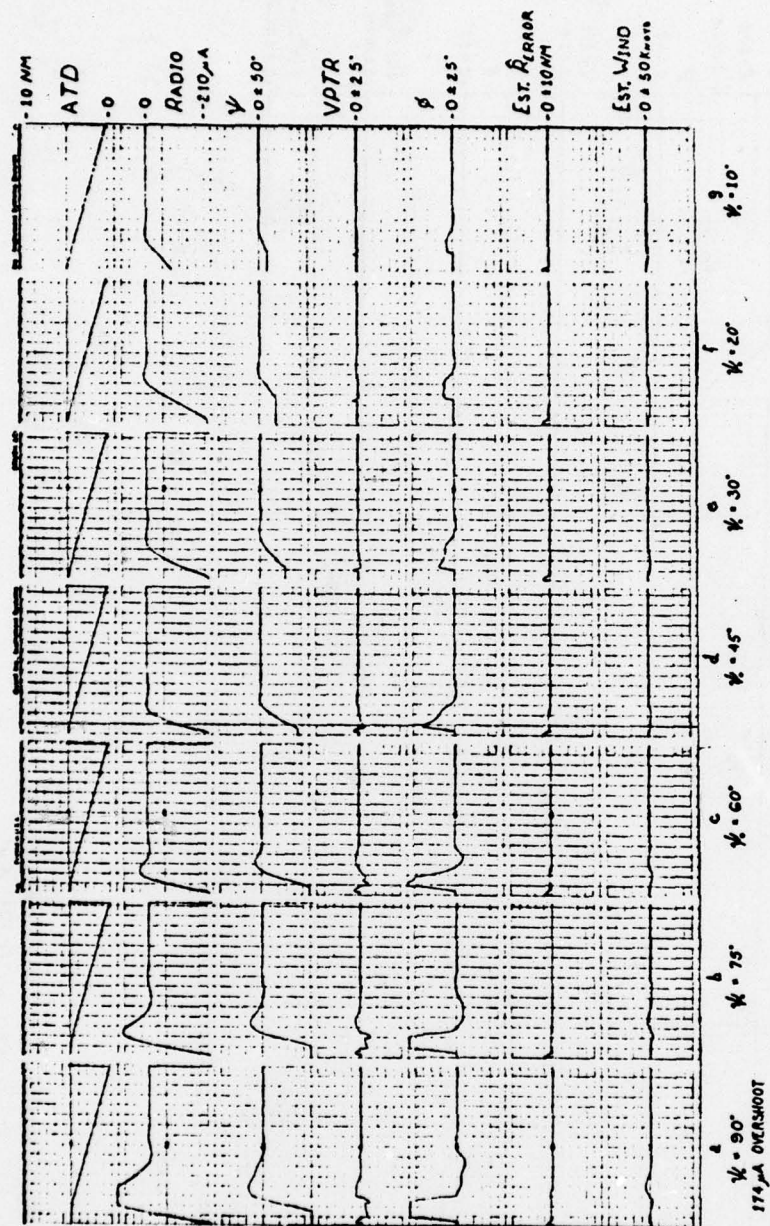


Figure 80. ILS Localizer Mode, Jet Transport, 125 Knots, ATD = 10 NM, $\psi = 45^\circ$, Wind, Radio Noise, ρ



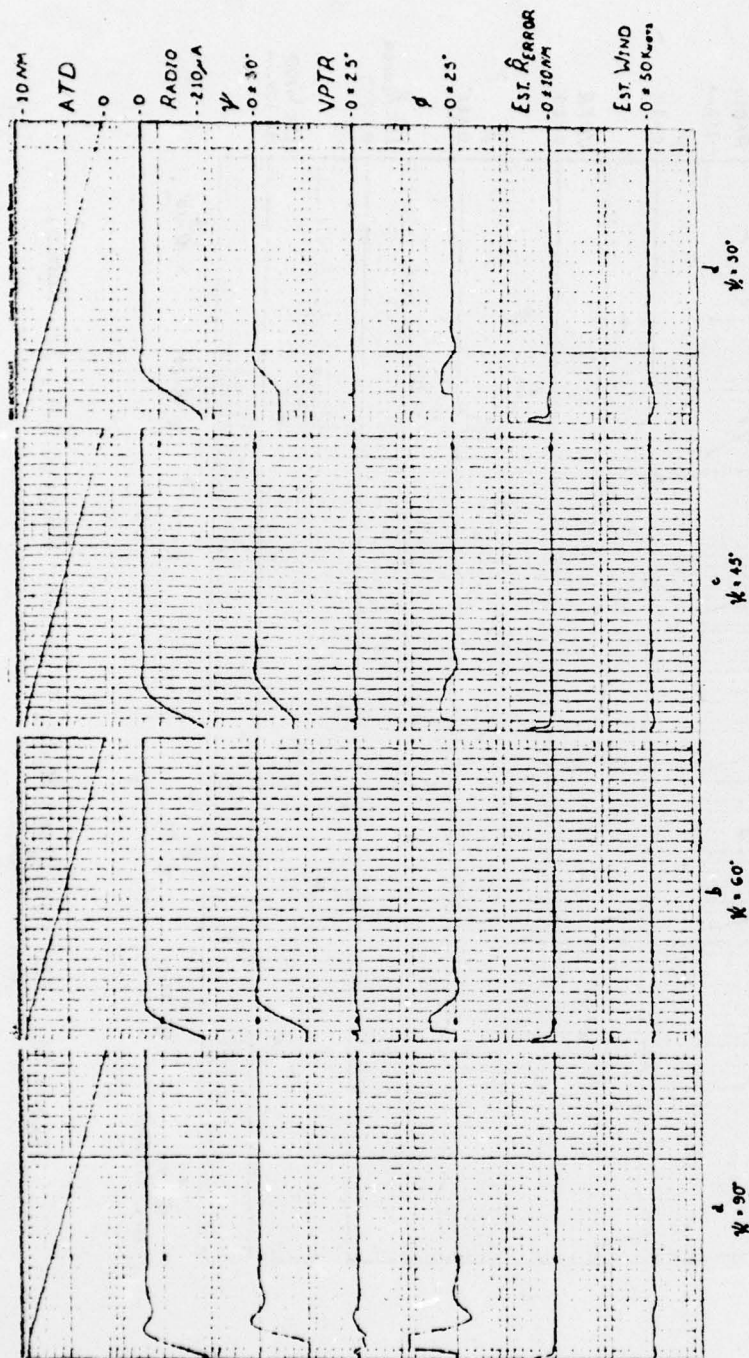


Figure 82. ILS Localizer Mode, Propeller Powered A/C, 120 Knots,
 $ATD = 10 \text{ NM}, \Delta \psi$

AD-A041 626

ROCKWELL INTERNATIONAL CEDAR RAPIDS IOWA COLLINS RA--ETC F/G 17/7
DIGITAL FLIGHT DIRECTOR COMPUTER.(U)
APR 75 J P DESMOND, G E FORQUER

F33657-73-C-0120

UNCLASSIFIED

ASD-TR-76-3

NL

3 of 4

ADA041626



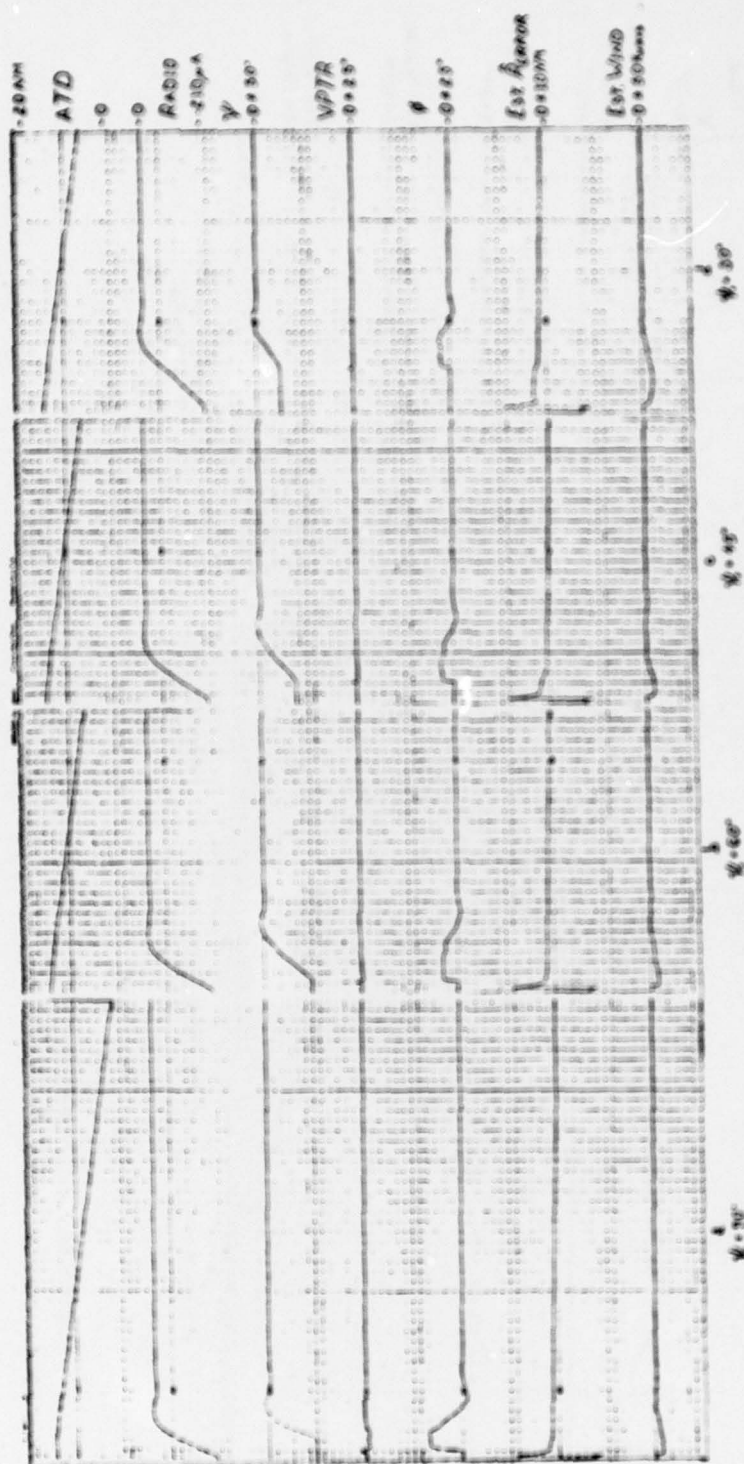


Figure 83. ILS Localizer Mode, Propeller Powered A/C, 120 Knots,
ATD = 15 NM, ΔY

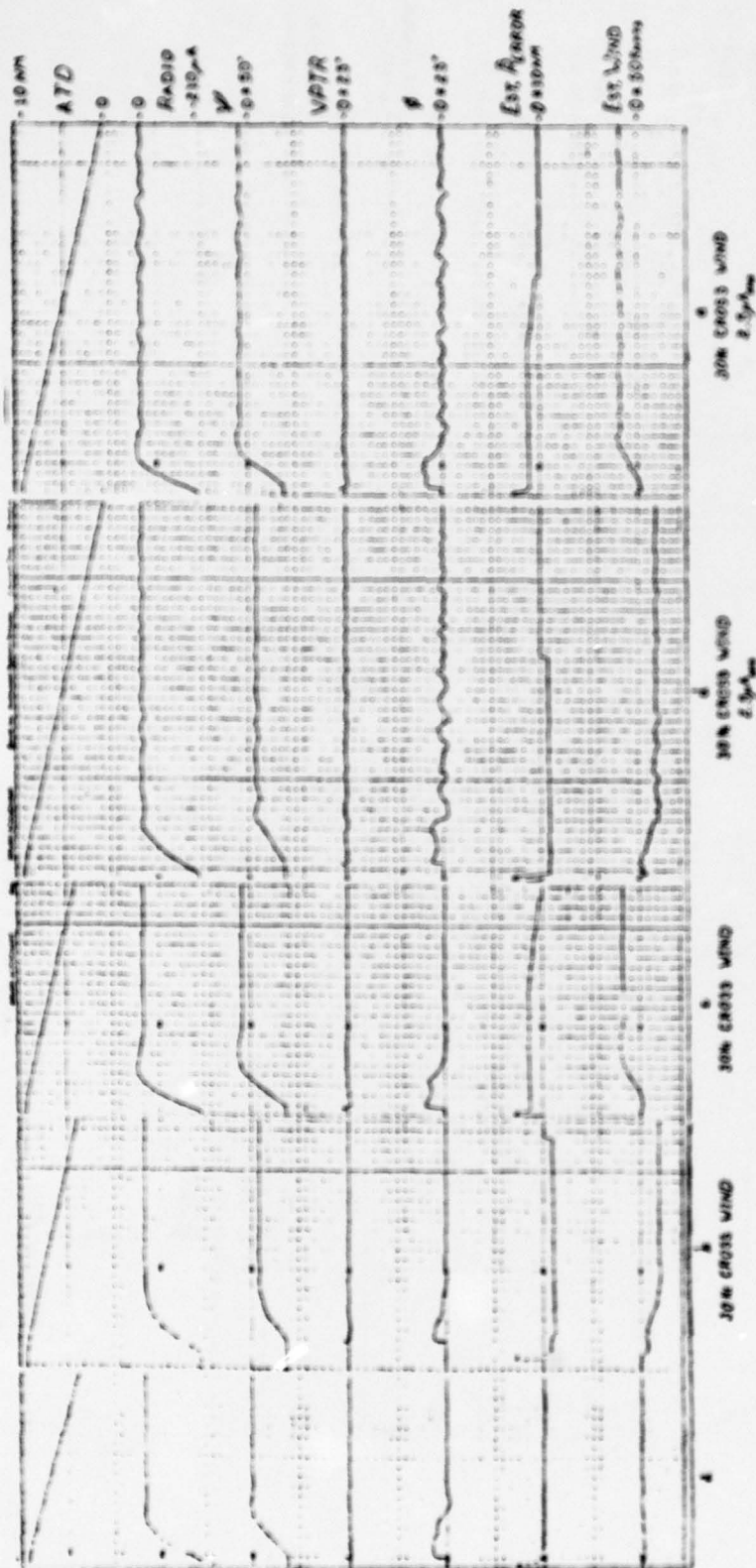


Figure 84. ILS Localizer Mode, Propeller Powered A/C, 120 Knots.
 ATD = 10 NM, $\psi = 45^\circ$, Wind, 2.5 μ A Radio Noise

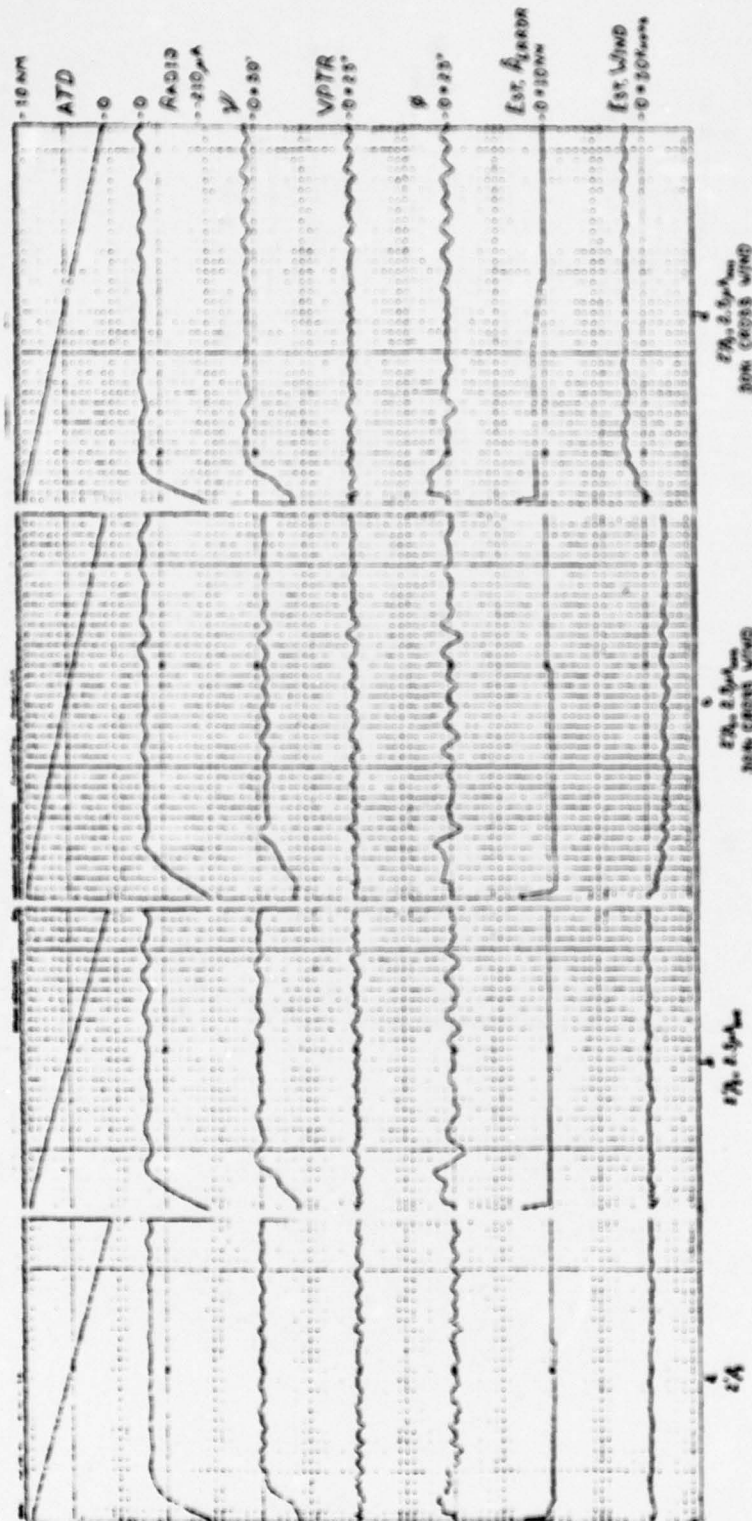


Figure 85. ILS Localizer Mode, Propeller Powered A/C, 120 Knots,
ATD = 10 NM, $\gamma = 45^\circ$, Wind, Radio Noise, P

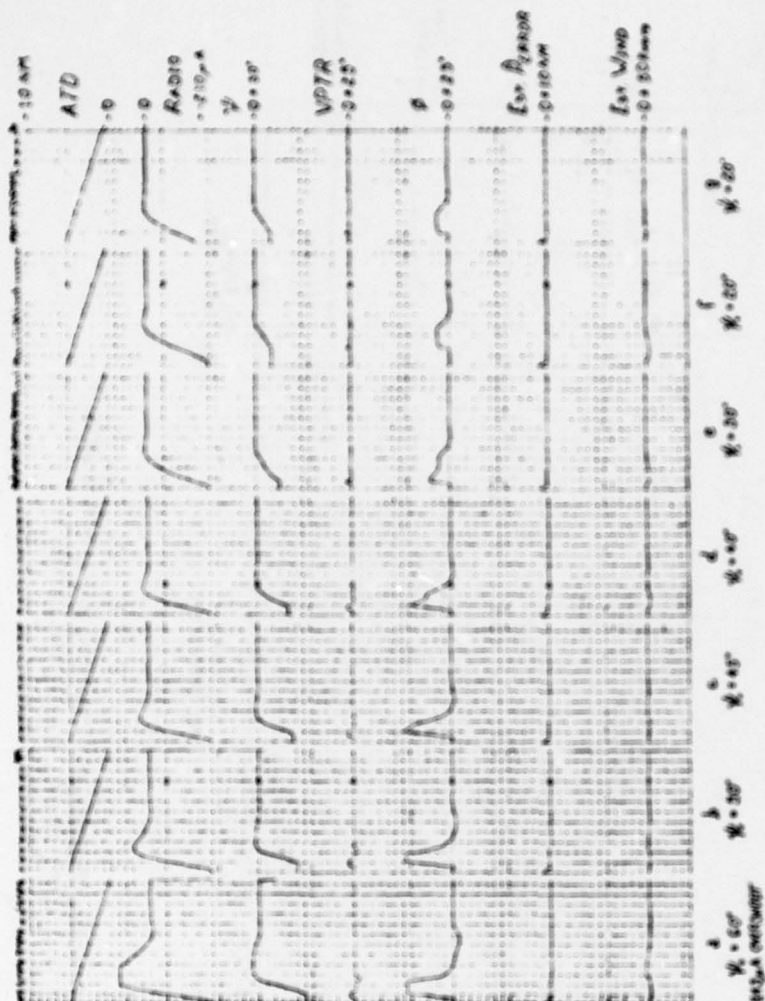


Figure 86. ILS Localizer Mode, Jet Aircraft, 150 Knots, ATD = 5 NM, 150 Kts

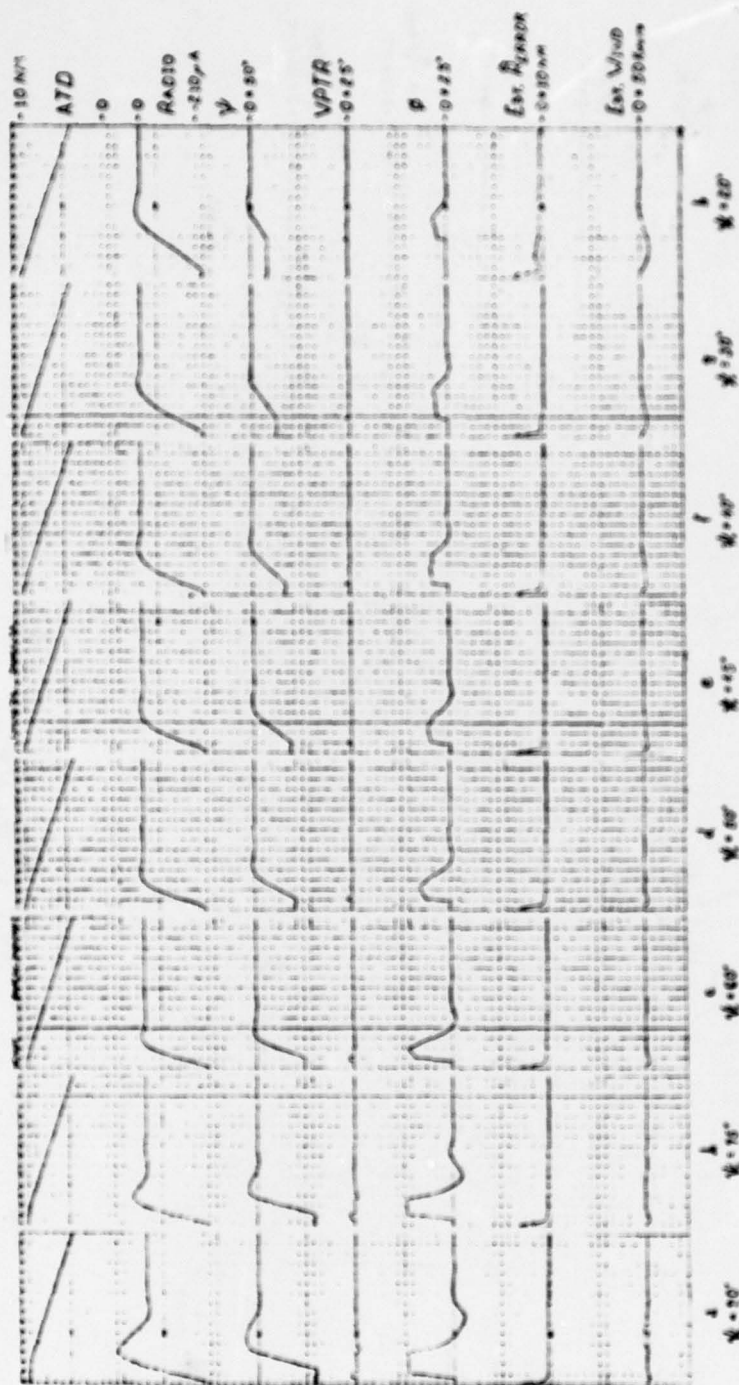


Figure 87. ILS Localizer Mode, Jet Aircraft, 150 Knots, ATD = 10 NM, ΔV

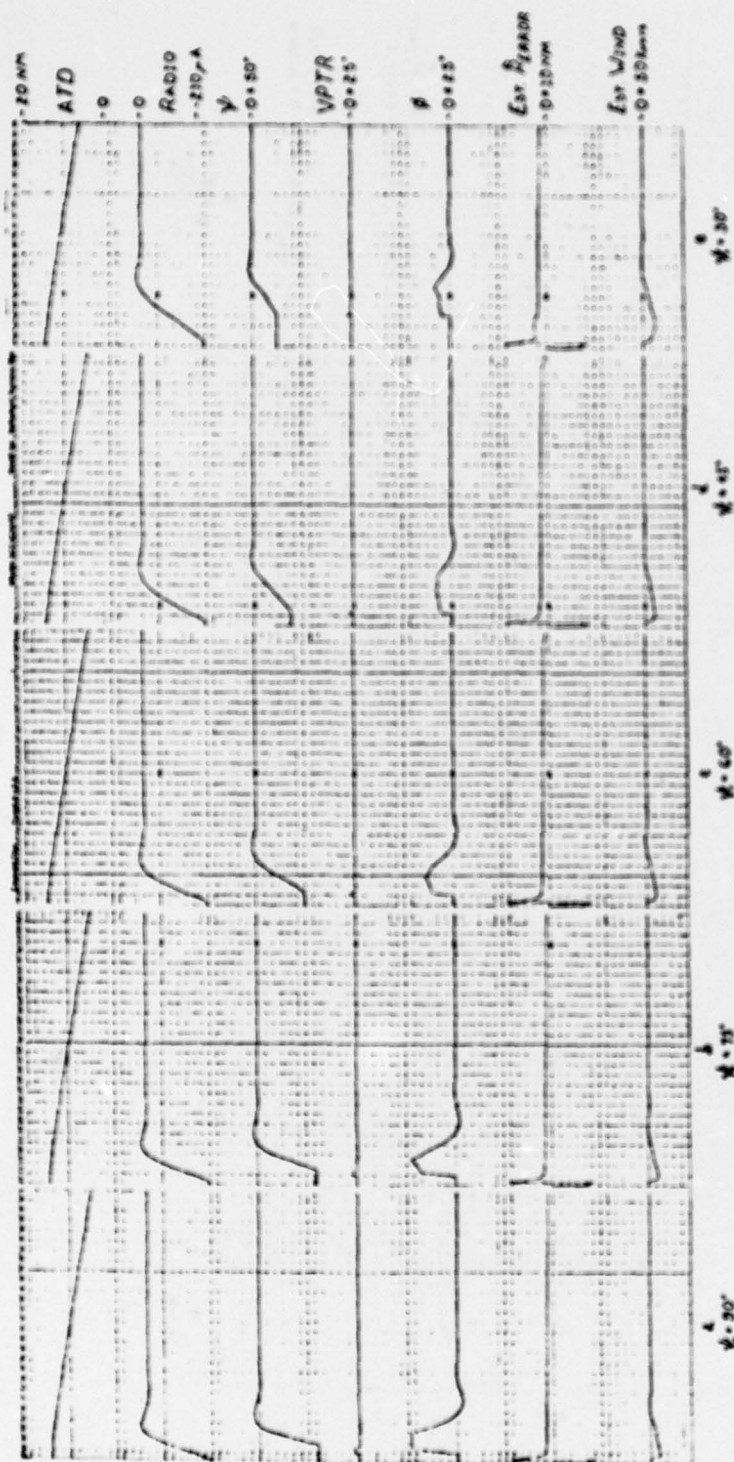


Figure 88. ILS Localizer Mode, Jet Aircraft, 150 Knots,
ATD = 15 NM, ΔY

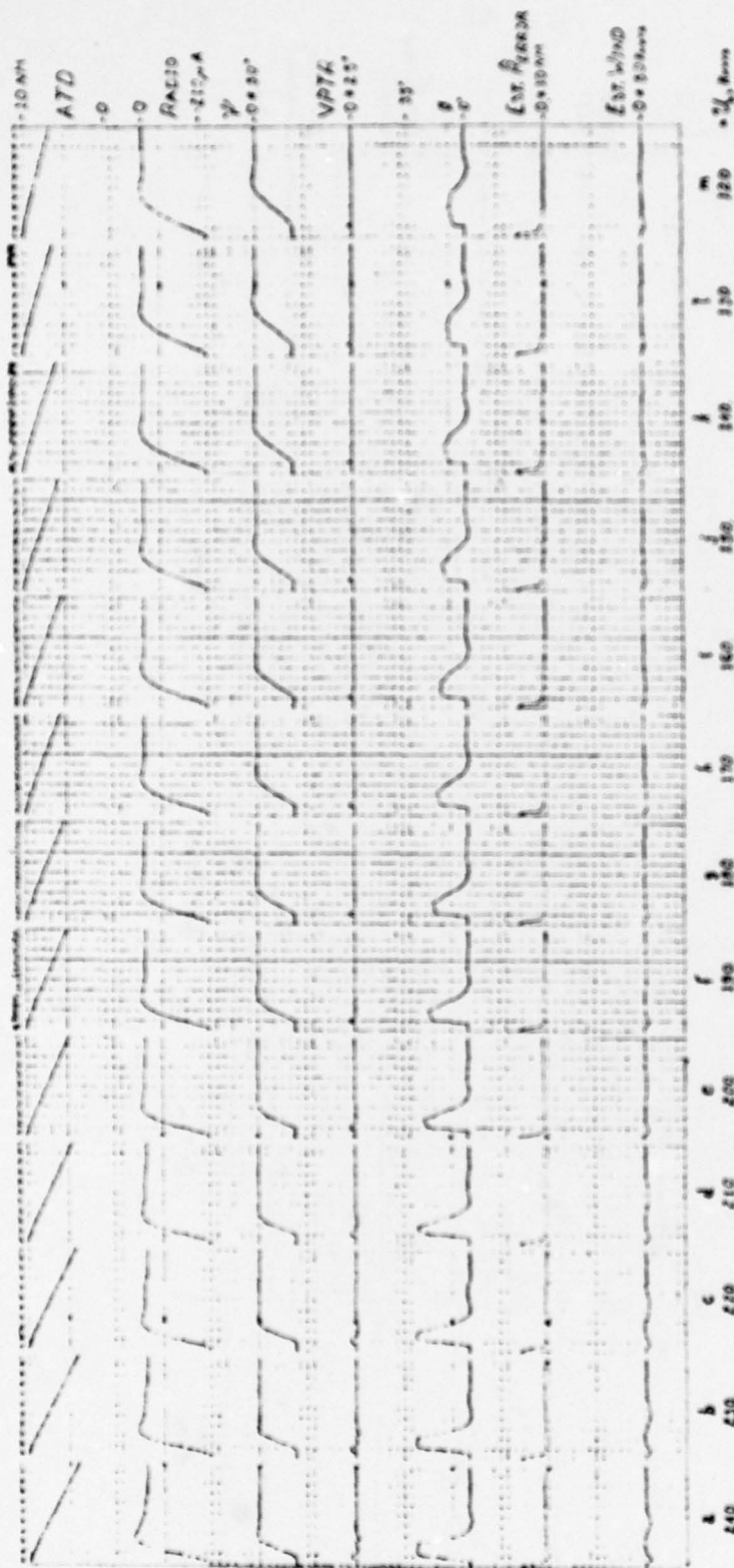


Figure 90. ILS Localizer Mode, Jet Aircraft, Δu
 ATD = 10 NM, $\psi = 45^\circ$



Figure 91. ILS Localizer Mode, Jet Aircraft, 150 Knots,
 ATD = 10 NM, $\psi = 45^\circ$, Wind, 2.5 μ A Radio Noise

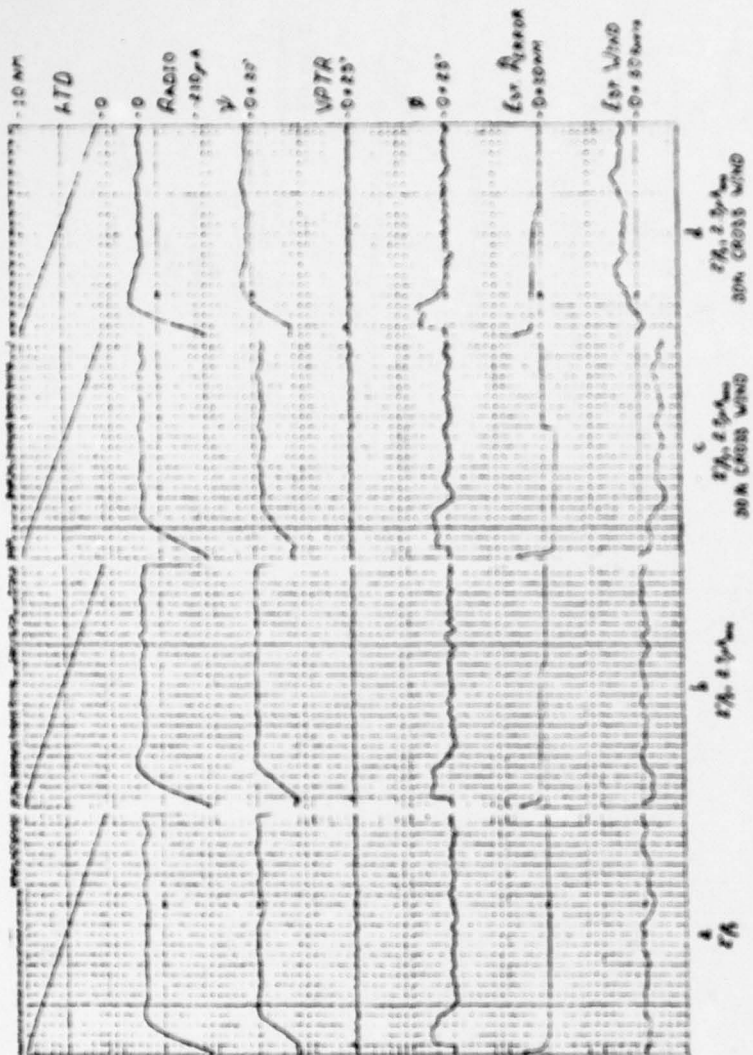


Figure 92. ILS Localizer Mode, Jet Aircraft, 150 Knots, ATD = 10 NM, $\gamma = 45^\circ$, Wind, Radio Noise, β

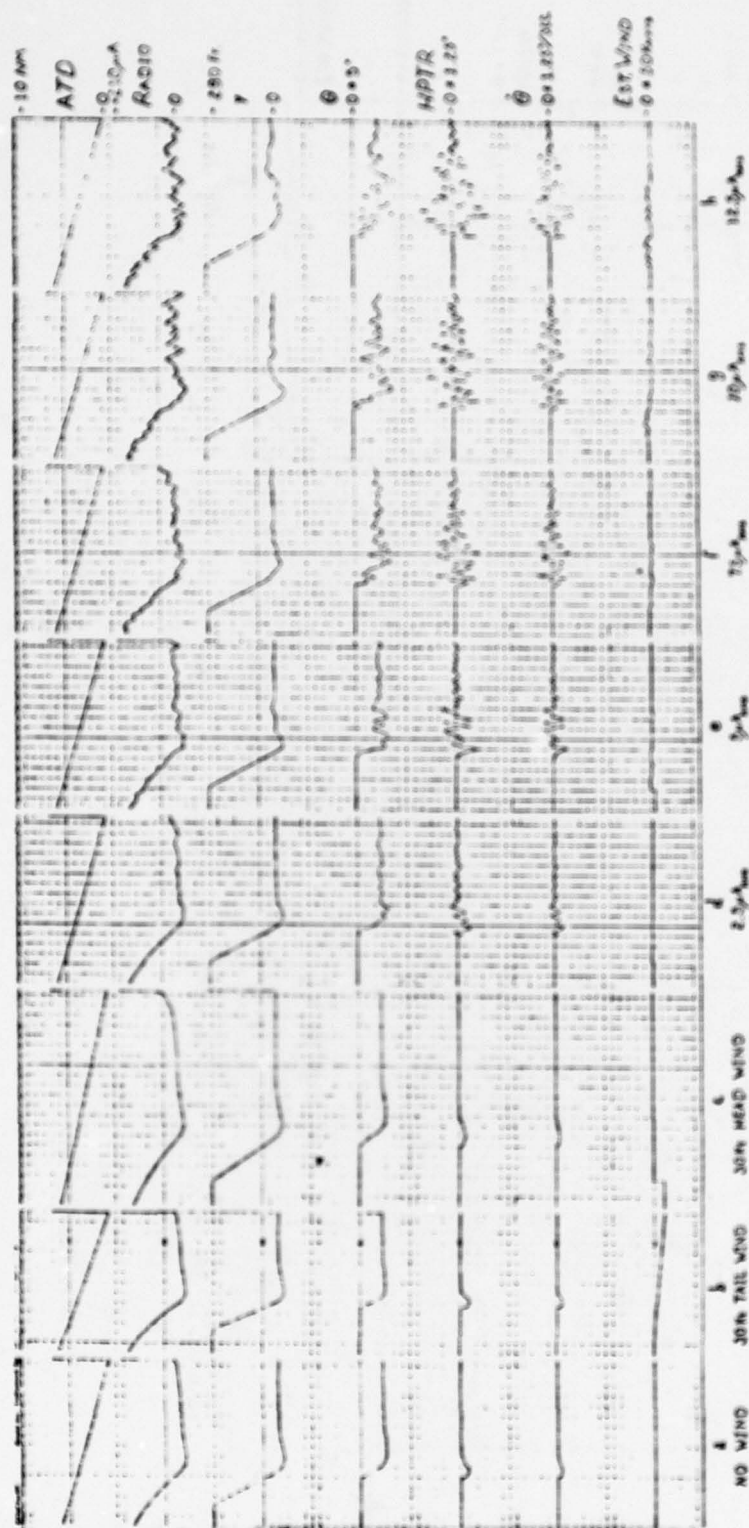


Figure 93. ILS Glideslope Mode - Jet Transport - 125 Knots - Wind - Radio Noise

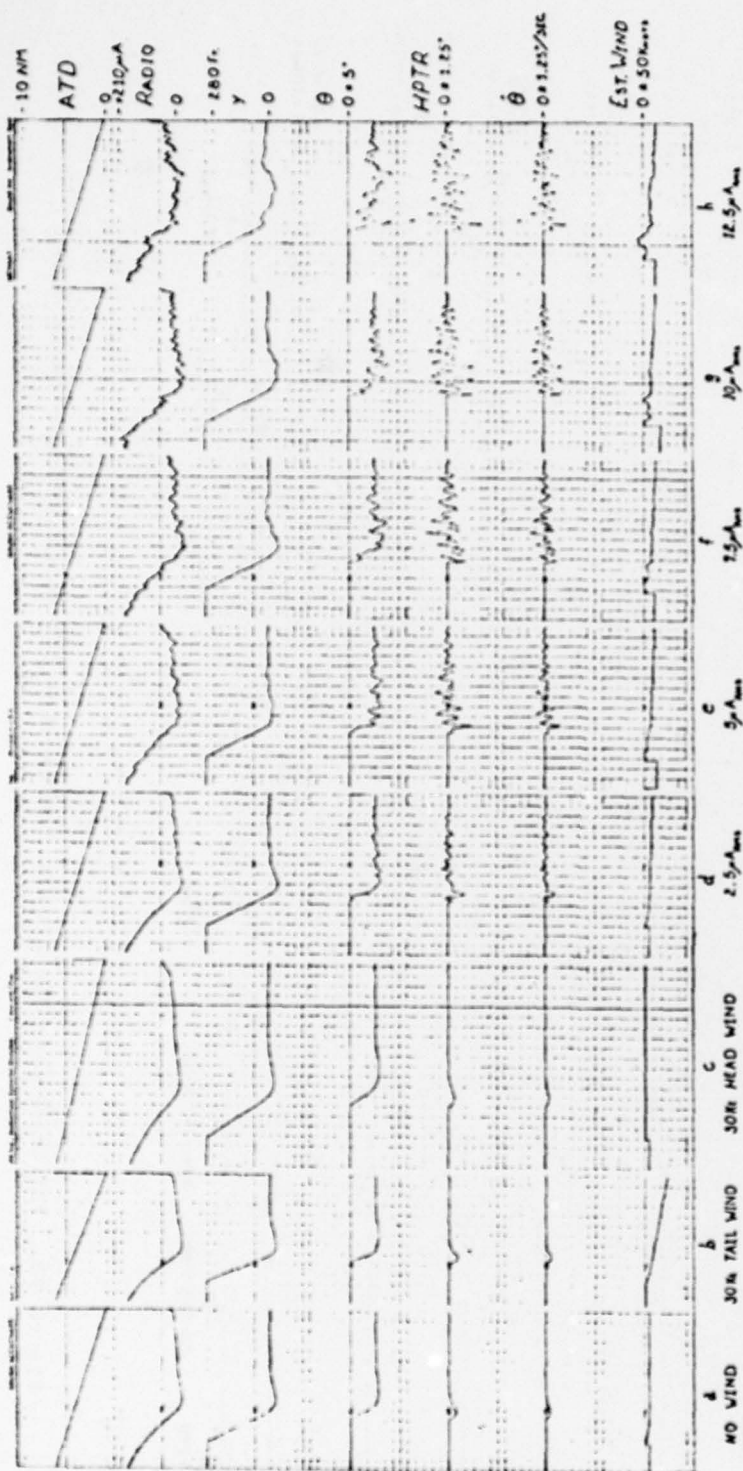


Figure 94. ILS Glideslope Mode - Propeller Powered A/C -
120 Knots - Wind - Radio Noise

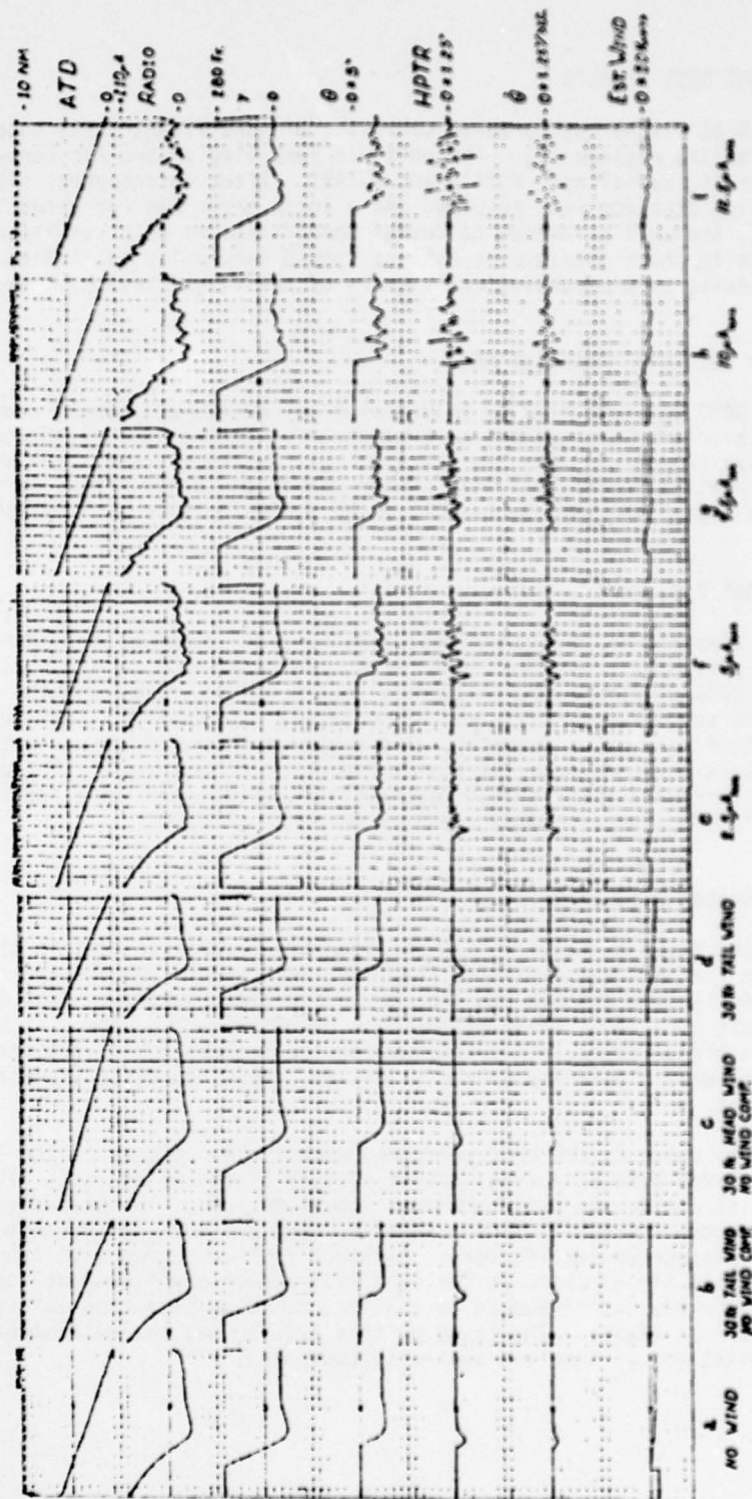


Figure 95. ILS Glideslope Mode - Jet Aircraft - 150 Knots - Wind - Radio Noise

SECTION VI

6.0 FLIGHT TEST RESULTS

The DFDC flight test program consisted of nine flight tests supported by Collins engineering. The unit was installed in two Air Force T-39 aircraft, serial nos. 603478 and 603481. After interconnect signal phasing problems were resolved and a logic error was corrected in the DFDC, the unit performed to design specifications with results comparable to those obtained at Collins hybrid simulation facilities. The following is a chronological summary of the flight test program.

July 23, 1974 Randolph AFB

The DFDC was installed in T-39 serial no. 603478 and ground checks of the aircraft interconnect were conducted. During this flight test, it was determined that the aircraft roll attitude inputs were phased incorrectly which prohibited proper operation in all flight modes. The bank signal phasing was corrected the following week at Randolph.

August 7,8, 1974

The aircraft was flown to Cedar Rapids, and a check of the interconnect was performed prior to flight tests. With the roll attitude corrected, the DFDC performed satisfactorily in the MAN HDG mode and the ILS track submode (localizer and glideslope). Attempts at localizer capture were not successful. The cause of this problem was determined to be a phase reversal in the course error interconnect. The phasing was corrected prior to subsequent flight tests. Due to a faulty aircraft DME receiver TACAN captures were not attempted.

September 5,6, 1974

Flight tests at Randolph AFB on September 5 revealed that in the ILS mode the DFDC was bypassing the localizer capture submode and proceeding directly from the MAN HDG submode to the track submode. This was the result of a logic error in the mode selection subroutine. It was possible to correct this problem for the September 6 flight test by reversing polarity on the glideslope input and pitch steering pointer output.

On September 6, the DFDC commanded acceptable captures of the ILS localizer beam at initial course cuts of 45 and 90 degrees. ILS tracking was successful in both localizer and glideslope. The DFDC also commanded successful TACAN captures; however track performance was not considered satisfactory. Although capture submodes of ILS and TACAN directed the aircraft to the beam with acceptable overshoot, the bank angle commanded tended to be shallow, normally less than 10° for 45° intercept angles. The cause of this problem was not determined due to limitations of time and available equipment.

October 3, 4, 1974

Prior to these flight tests, one programmable read only memory located on the A6 card assembly was reprogrammed to reverse glide-slope phasing. This corrected the mode selection logic and permitted the DFDC to enter the capture routines. Attempts were made prior to flight tests by the Air Force and Collins to obtain an airborne recorder for troubleshooting. A suitable recorder was not available and it was therefore not possible to monitor DFDC inputs during the tests.

The DFDC operated successfully in all modes during flight tests conducted on both days. TACAN tests included low and high speed captures at large and small intercept angles. The capability of the unit to select a new outbound radial in TACAN overstation was demonstrated. ILS localizer and glideslope captures at low speed shallow intercept angles and high speed large intercept angles were also demonstrated. In the ILS track mode, radar altitude gain programming performed to design specifications.

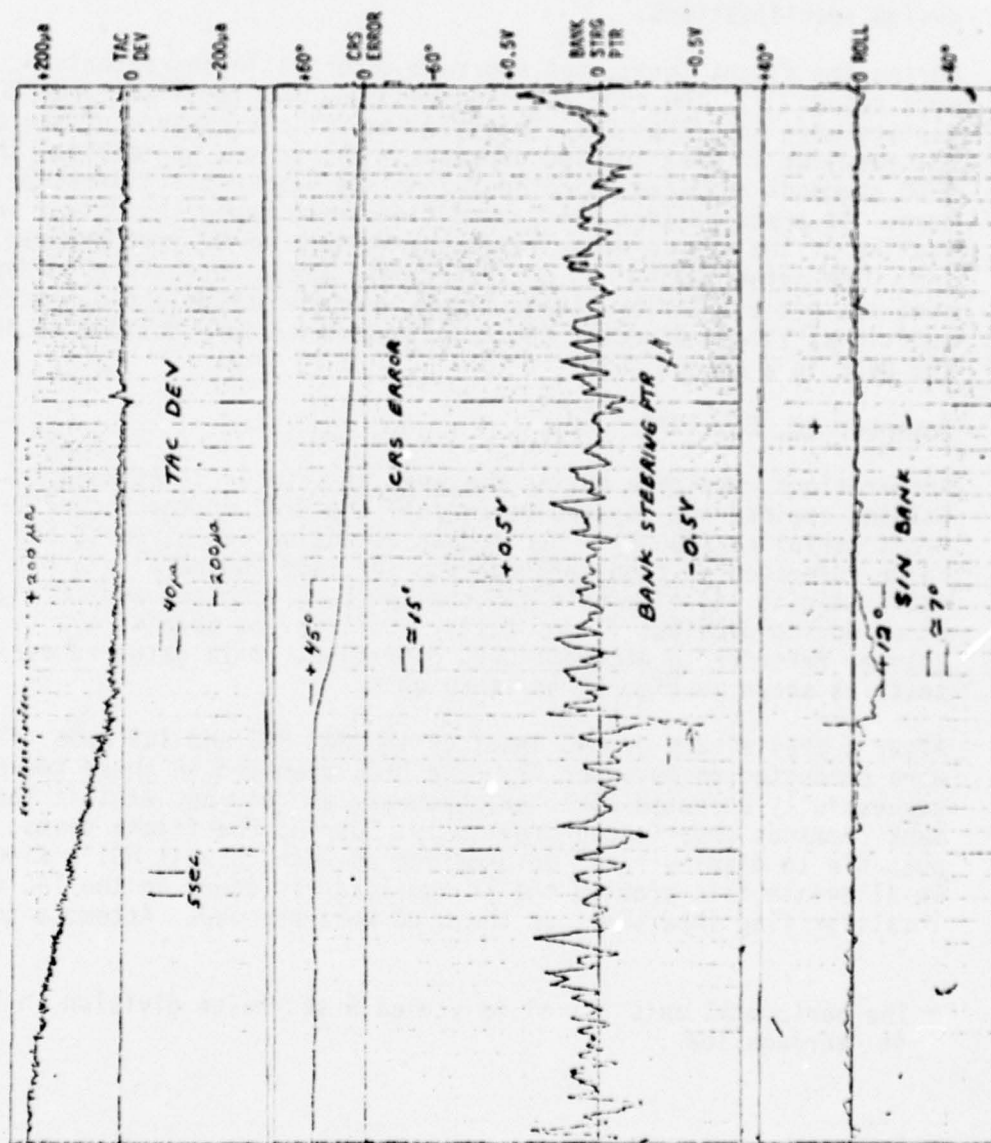
During the flight tests, two apparent undesirable characteristics were noted: 1) The bank angle commands during localizer and TACAN capture were less than 8° . This did not inhibit capture of the beam, but resulted in long shallow captures. 2) On two approaches, the DFDC commands resulted in "S"ing along the beam until glideslope capture. It appeared that these conditions were caused by either incorrect scaling of the course error reference voltage or by improper radio loading. The actual cause was not determined since adequate recording equipment was not available. The aircraft project schedule did not permit additional flight director tests. It was therefore decided to install the DFDC in a second T-39.

November 22, 23, 1974

Preparations were made during the week of October 21 at Randolph AFB with Collins engineering support to install the DFDC in the second T-39 aircraft serial no. 603481. Particular attention was given to the course error reference voltage derived from the 115V 400 HZ aircraft power and radio loading to eliminate the shallow bank commands and "S"ing experienced during previous flight tests. Collins was able to supply an eight channel recorder to document DFDC operation. Data gathered during the tests is shown in Figures 96 thru 99 *.

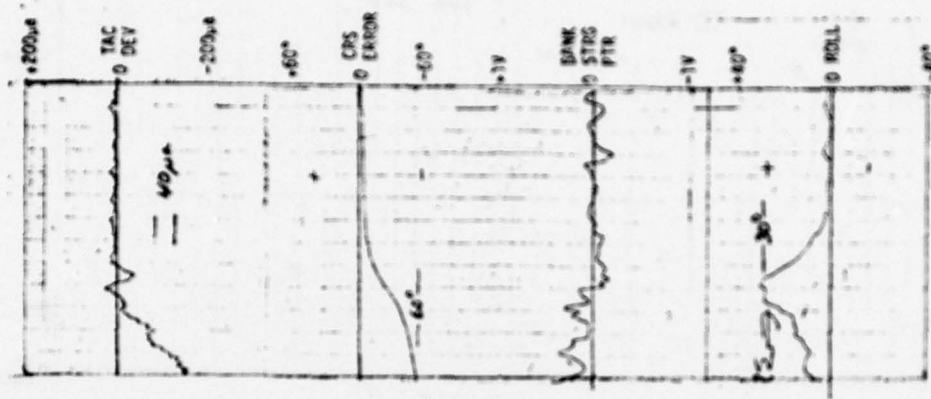
After a preliminary ground check of the MAN HDG and ILS mode, flight tests were conducted on November 22. The DFDC operated in these two modes, successfully executed circular captures, and did not exhibit the shallow bank commands experienced previously. During the flight tests, it was not possible to display radio information on both cockpit HSI's simultaneously. To alleviate this problem the 1K ohm load resistors on the TAC dev and localizer flag inputs within the DFDC were removed. Attempts to capture

* The horizontal axis (time) is scaled 5 sec/major division in Figures 96 through 108 .

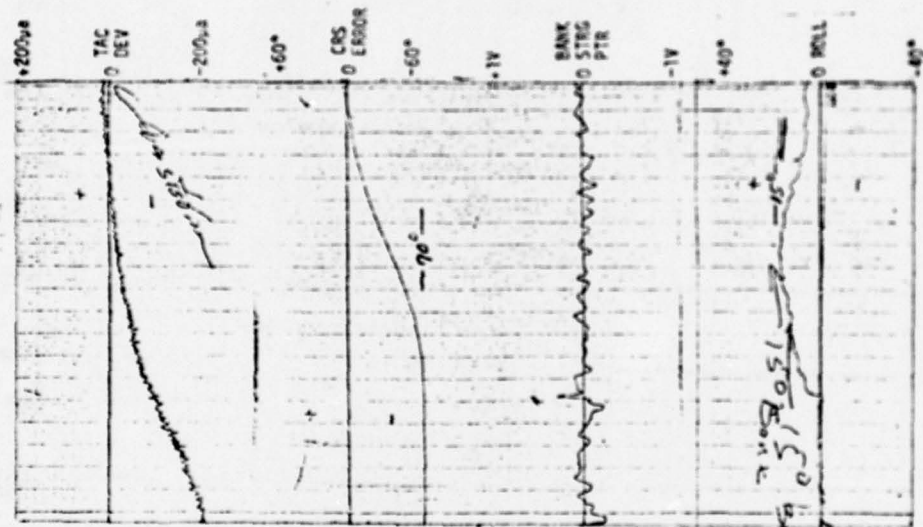


Altitude: 3000 ft
 Initial Course Error: 45°
 Distance to Station at Beam Intercept: 70 nm

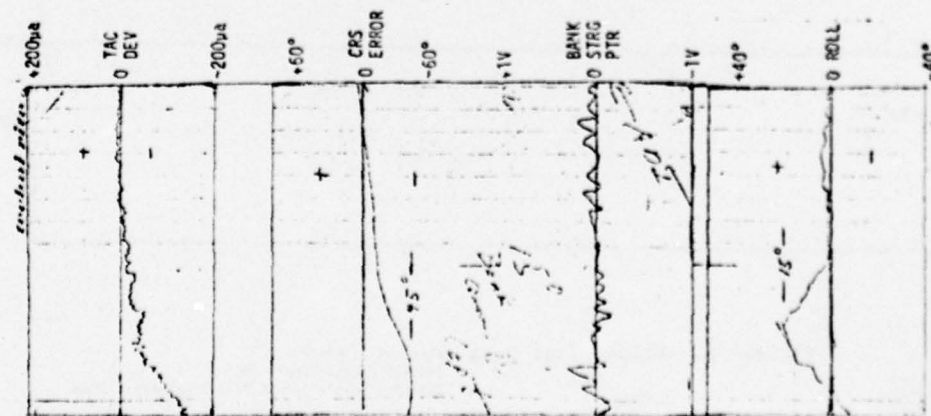
Figure 96. Flight Test Data - TACAN



Airspeed: 400kts
 Initial Course Error: 45°
 Distance to Station at Intersection: 40nm



Airspeed: 400kts
 Initial Course Error: 90°
 Distance to Station at Intersection: 60nm



Airspeed: 400kts
 Initial Course Error: 90°
 Distance to Station at Intersection: 60nm

Figure 97. Flight Test Data - TACAN

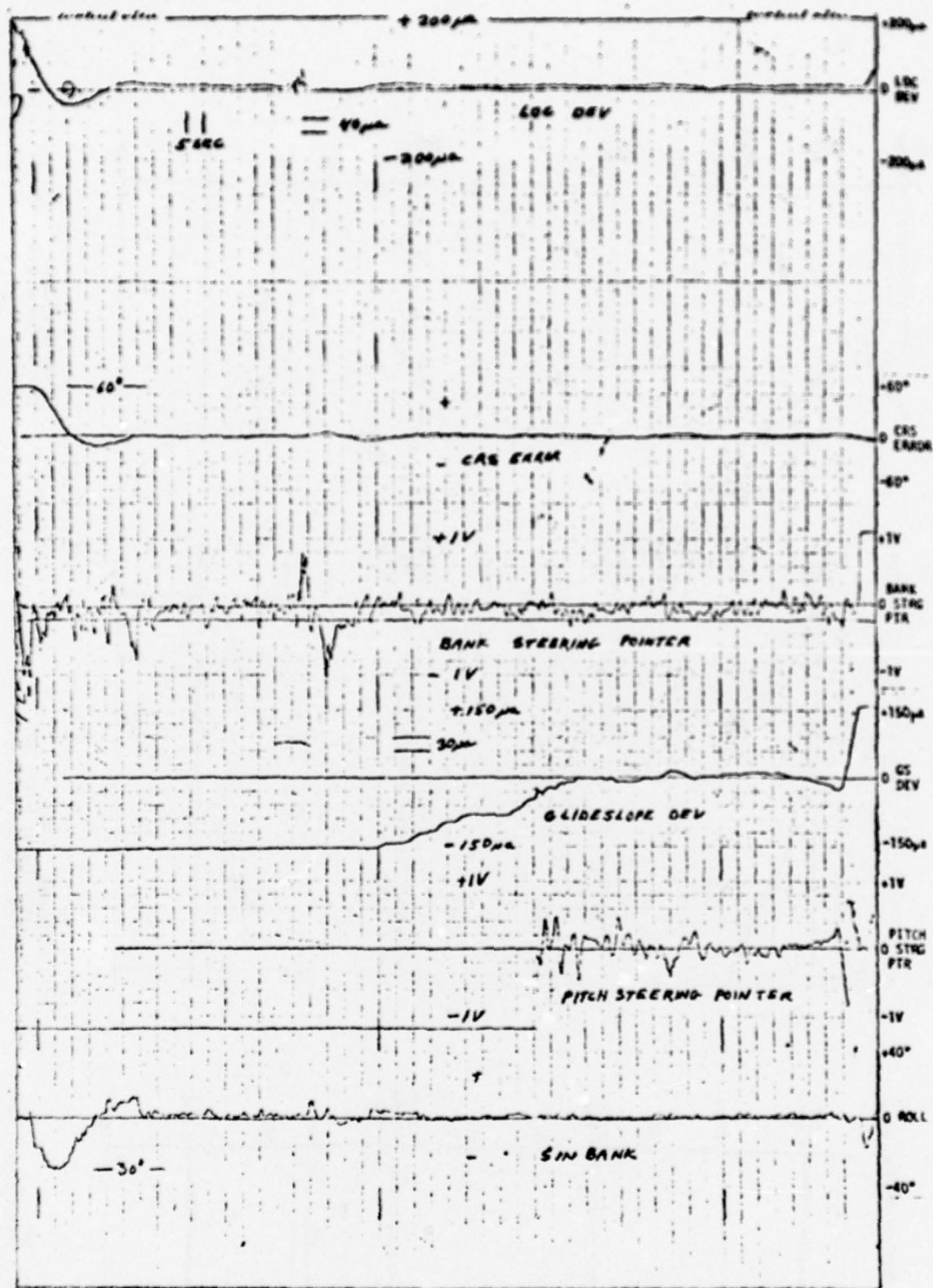


Figure 98. Flight Test Data
 ILS
 194
 -Altitude: 1000ft
 Initial Course Error: 60°
 Distance to Station at Beam Intercept: 18nm

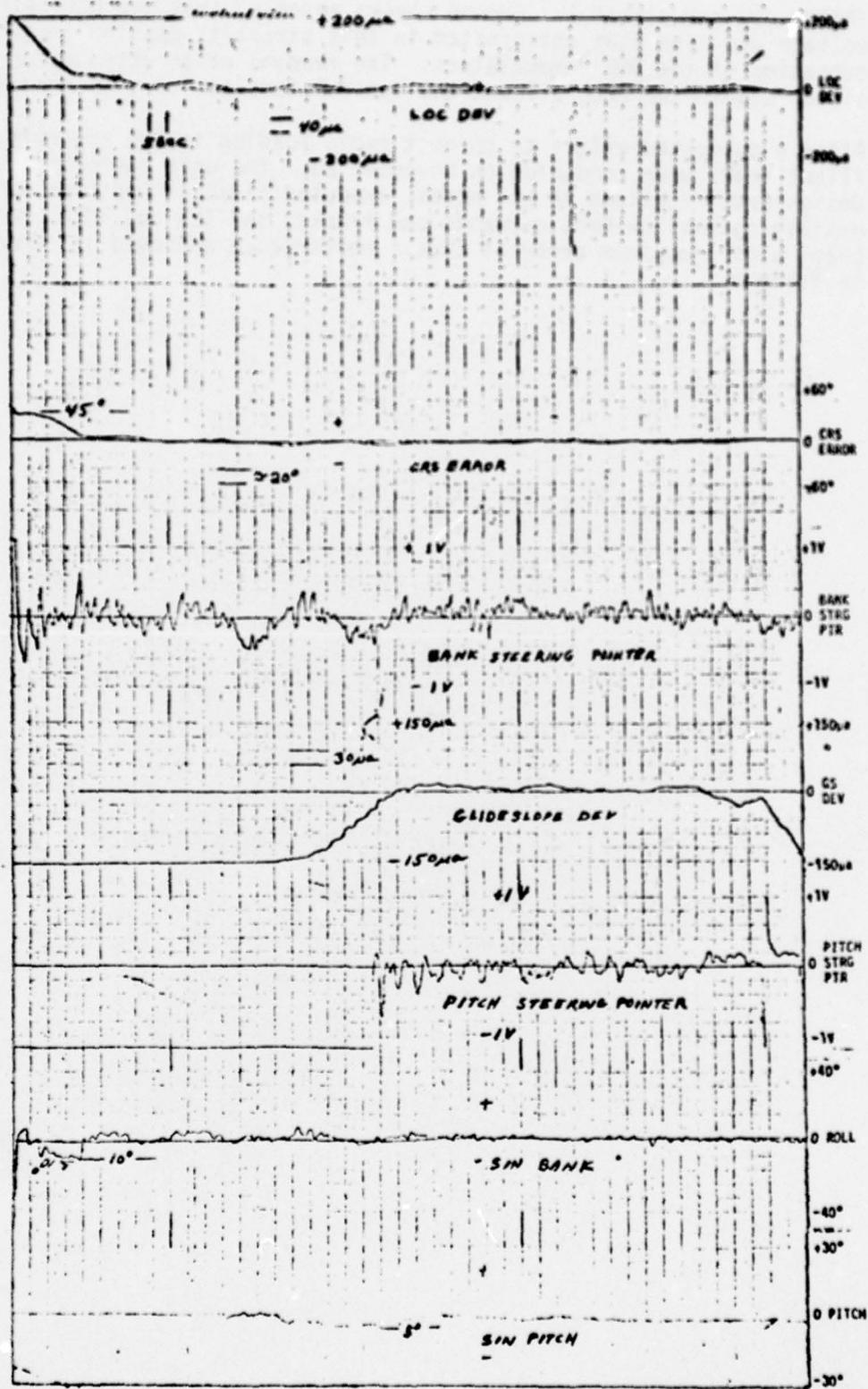


Figure 99. Flight Test Data - 195 ILS
 Altitude: 18000 ft
 Initial Course Error: 35°
 Distance to Station at Beam Intercept: 15 nm

TACAN were unsuccessful. Ground checks revealed that the DME reference voltage was less than anticipated in this aircraft, causing improper operation of the DME demodulators. The removal of an attenuation resistor within the DFDC allowed the demodulators to function normally.

After a concerted effort to correct radio loading in the aircraft, flight tests were conducted on November 23. The unit performed to design specifications in all modes, executed circular captures without exhibiting shallow bank commands and tracked the ILS and TACAN radio beams with a maximum error at 20ua. Performance achieved is summarized in Table 9 .

TABLE 9. DFDC

PERFORMANCE RESULTS T-39 FLIGHT TESTS

Distance to Station at Beam Intercept		Initial Course Error	Airspeed	Commanded Bank	Overshoot	Maximum Track Error
TACAN	1) 70 nm inbound	45°	300 kts	12°	0	15 µa
	2) 40 nm inbound	45°	400 kts	15°	0	10 µa
	3) 60 nm inbound	90°	400 kts	15°	0	*
	4) outbound	60°	400 kts	30°	0	*
ILS	1) 15 nm	30°	180 kts	10°	0	15 µa LOC 20 µa GS
	2) 15 nm	45°	180 kts	20°	30 µa	10 µa LOC 10 µa GS
	3) 15 nm	60°	180 kts	30° (Bank Limit)	40 µa	15 µa LOC 20 µa GS
	4) 15 nm	90°	180 kts	30° (Bank Limit)	100 µa	*
	5) 10 nm	60°	180 kts	30° (Bank Limit)	30 µa	*

* Data Not Collected.

November 22, 23, 1974

Due to time limitations,
heading was changed after
beam intercept to obtain
additional capture data.

March 3-10, 1975

Engineering support and recording equipment were provided at Randolph AFB during the week of March 3 for verification and documentation of DFDC in-flight performance. During this week; two aircraft problems, which caused intermittent system operation, were located. The roll gyro behaved erratically around zero with output deviations of 3 to 10° from true attitude. This problem was reduced by replacing the roll gyro, but was not completely eliminated. A loose localizer channel interconnect wire was located in the project junction box. This problem caused intermittent mode switch inhibiting, in the ILS submodes. The loose connection was corrected and normal ILS operation resumed.

The DFDC performed to design specifications as shown in Figures 100 and 101* during most captures. However, an apparent instability in the system caused the DFDC to command an "S"ing down the localizer beam during several capture attempts.

The computer was returned to Collins and interfaced to the hybrid simulation facility. A concerted effort was undertaken to recreate the characteristics observed during flight tests. A thorough examination of the data obtained from the flight tests revealed that the instability or "S"ing occurred at times when the pilot was not maintaining the bank steering pointer at zero. By approximating this condition, the instability noted in flight was evident under simulation. The effects of this phenomena were minimized by modifying the localizer track law. The track law modification consisted of reducing bank command due to beam deviation which increased bank command response as a function of beam rate.

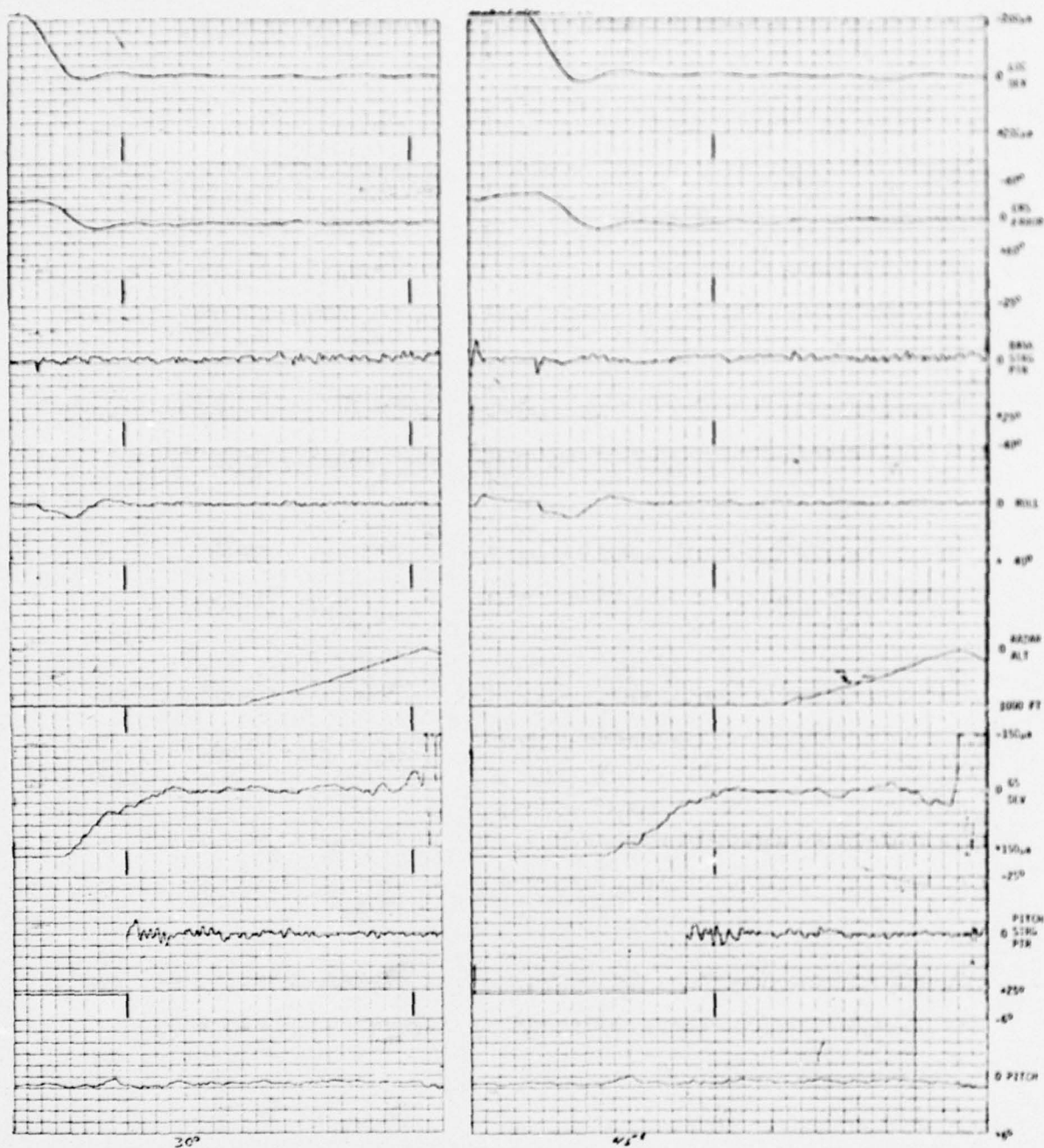
This modified track law was implemented in the DFDC by changing one constant which required reprogramming four PROMS. Further simulation was conducted to verify the increased stability and to insure that the modification did not result in decreased performance.

March 18-21, 1975

The DFDC was returned to Randolph with the modified program card installed and flight tests were conducted with recording equipment for four additional days. The flight tests demonstrated successful operation and stability in all modes confirming the simulator results obtained. Data collected during this last week of flight tests is shown in Figures 102 through 108 *. Performance attained during the March flight tests is summarized in Table 10 .

System performance is best illustrated in Figures 96 through 101 and 108 where the bank steering pointer was maintained near zero during capture and track.

* The horizontal axis (time) is scaled 5 sec/major division in Figures 96 through 108.



Airspeed: 170 kts
Intercept Angle: 30°
Range to Ray at Intercept: 8 mi

10 SEC/MAJOR DIV

Airspeed: 170 kts
Intercept Angle: 45°
Range to Ray at Intercept: 10 mi

Figure 100. Flight Test Data - ILS

Date: 3/5/75



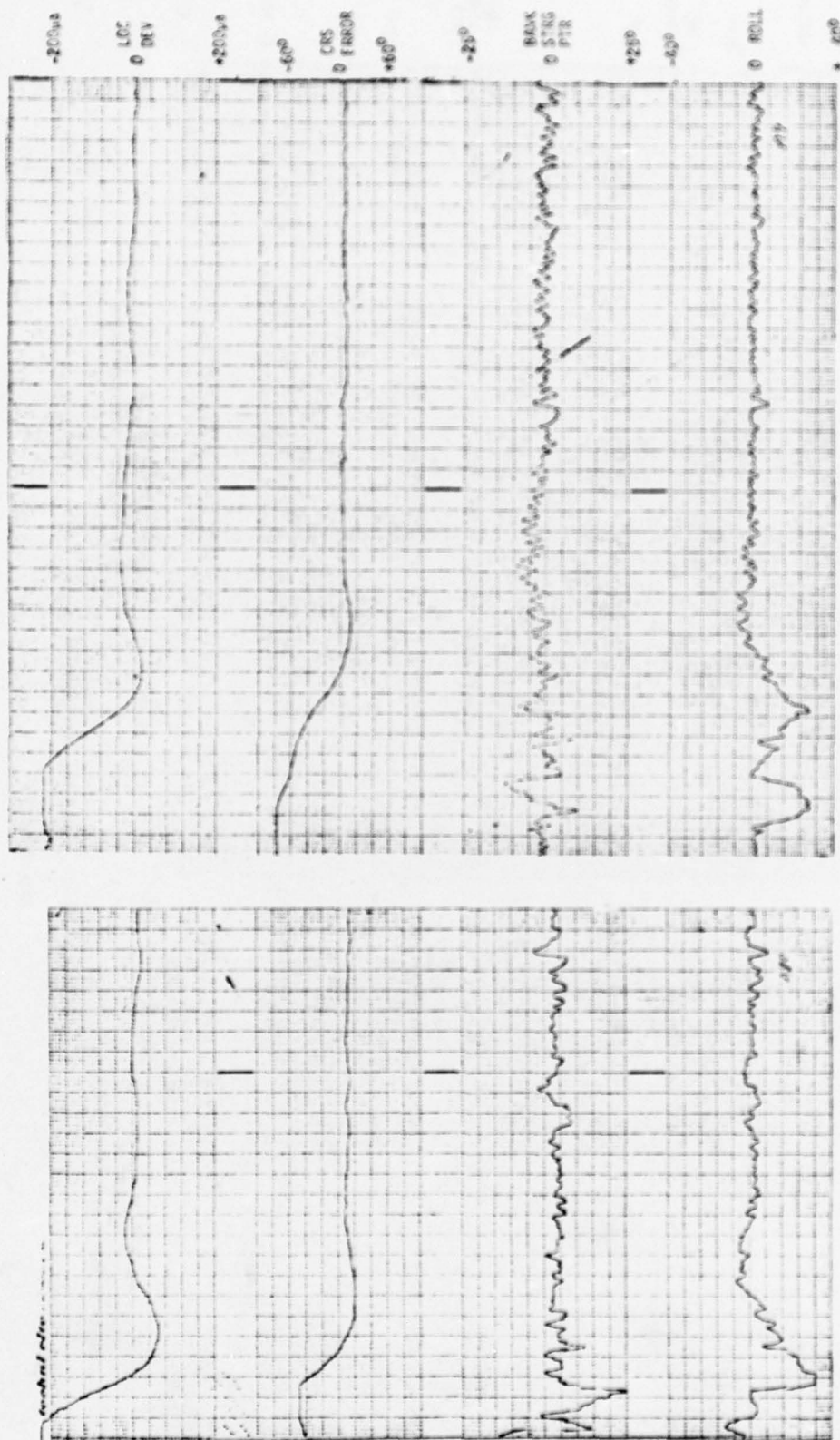
Airspeed: 170 Kts
Intercept Angle: 60°
Range to Ray at Intercept: 15 mi

10 SEC/Major DIV

Airspeed: 170 Kts
Intercept Angle: 90°
Range to Ray at Intercept: 18 mi

Date: 4/5/75

Figure 101. Flight Test Data - ILS

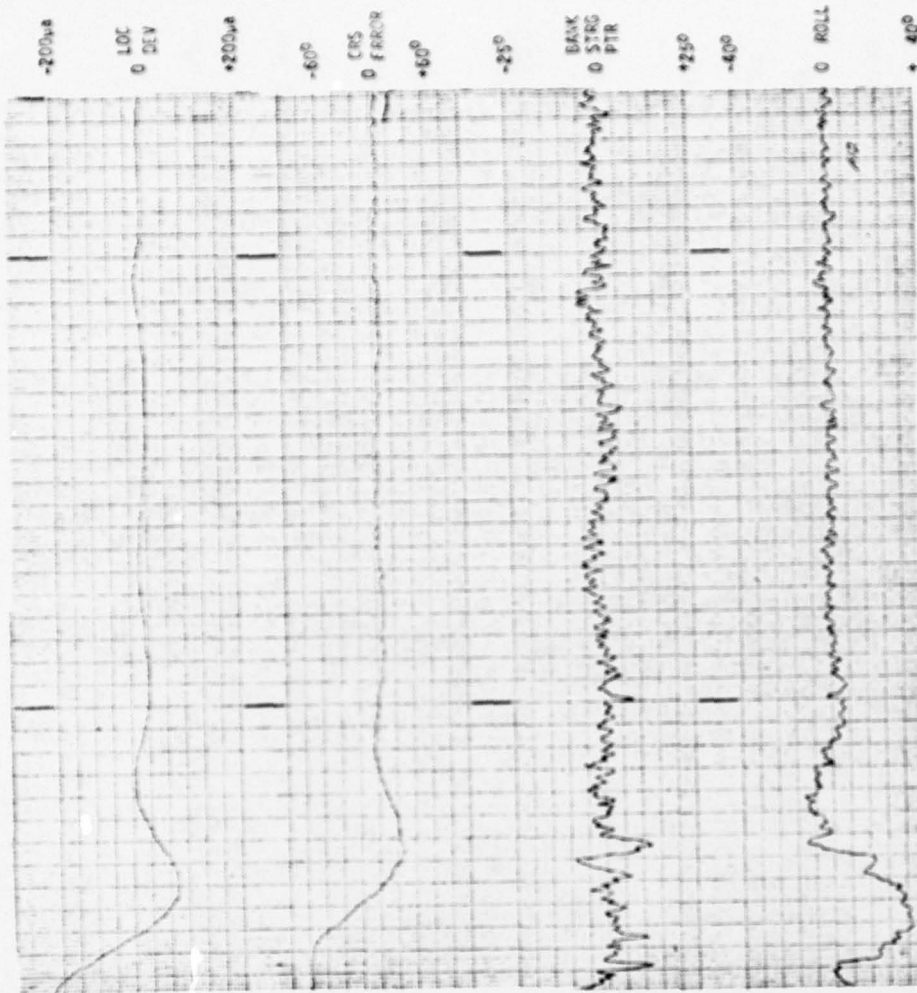


Airspeed: 170 Kts
Intercept Angle: 45°
Range to Ray at Intercept: 15 mi

Airspeed: 170 Kts
Intercept Angle: 30°
Range to Ray at Intercept: 8 mi

Date: 3/10/55

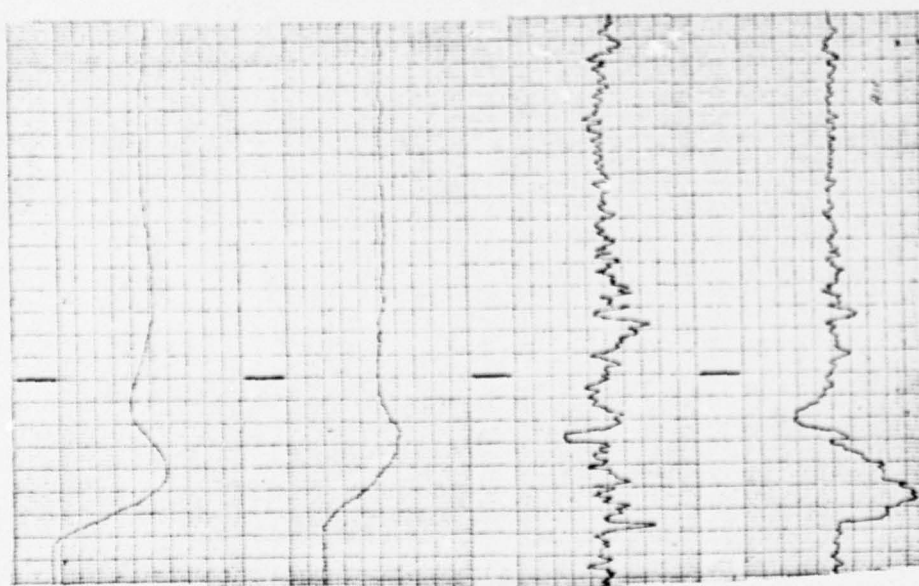
Figure 102. Flight Test Data - ILS



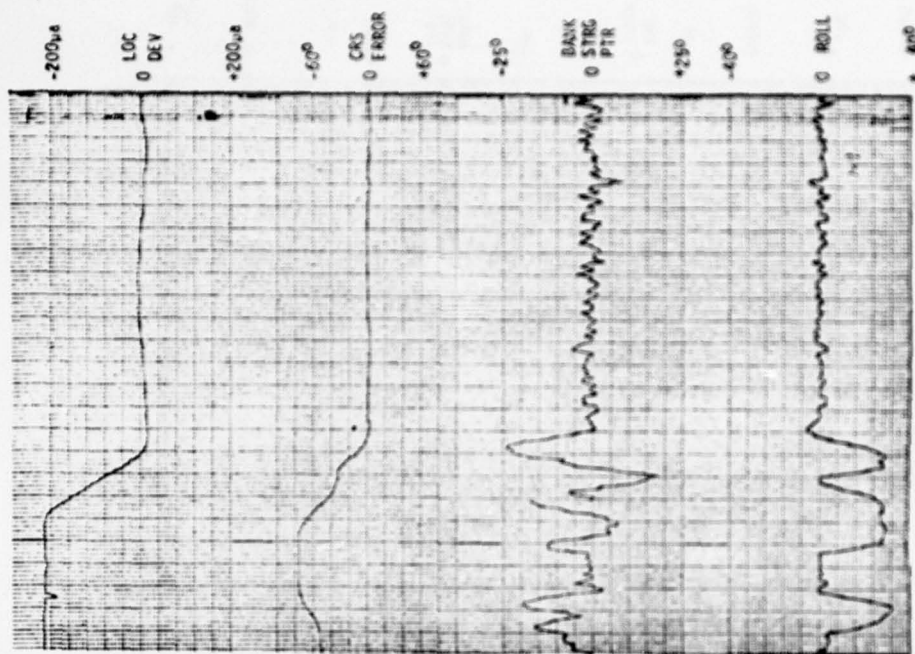
Airspeed: 170kts
Intercept Angle: 90°
Range to Ray at Intercept: 18mi

Figure 103. Flight Test Data - ILS

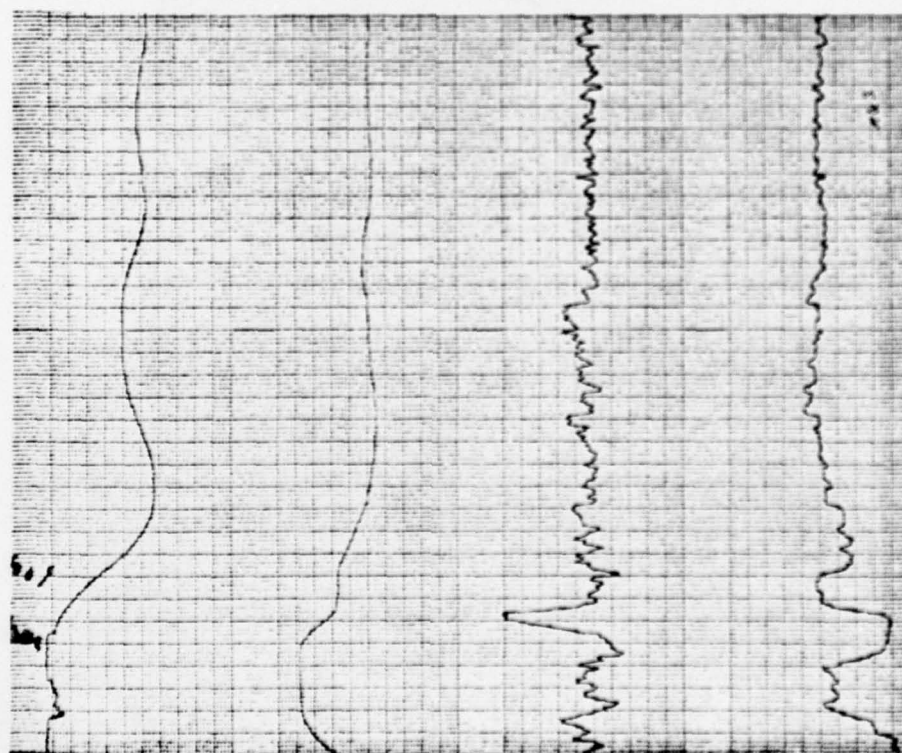
Date: 3/20/75



Airspeed: 170kts
Intercept Angle: 60°
Range to Ray at Intercept: 10mi



Airspeed: 200 Kt
Intercept Angle: 45°
Range to Hwy at Intercept: 8 mi



Airspeed: 200 Kt
Intercept Angle: 30°
Range to Hwy at Intercept: 8 mi

Figure 104. Flight Test Data - ILS

Date: 3/19/75

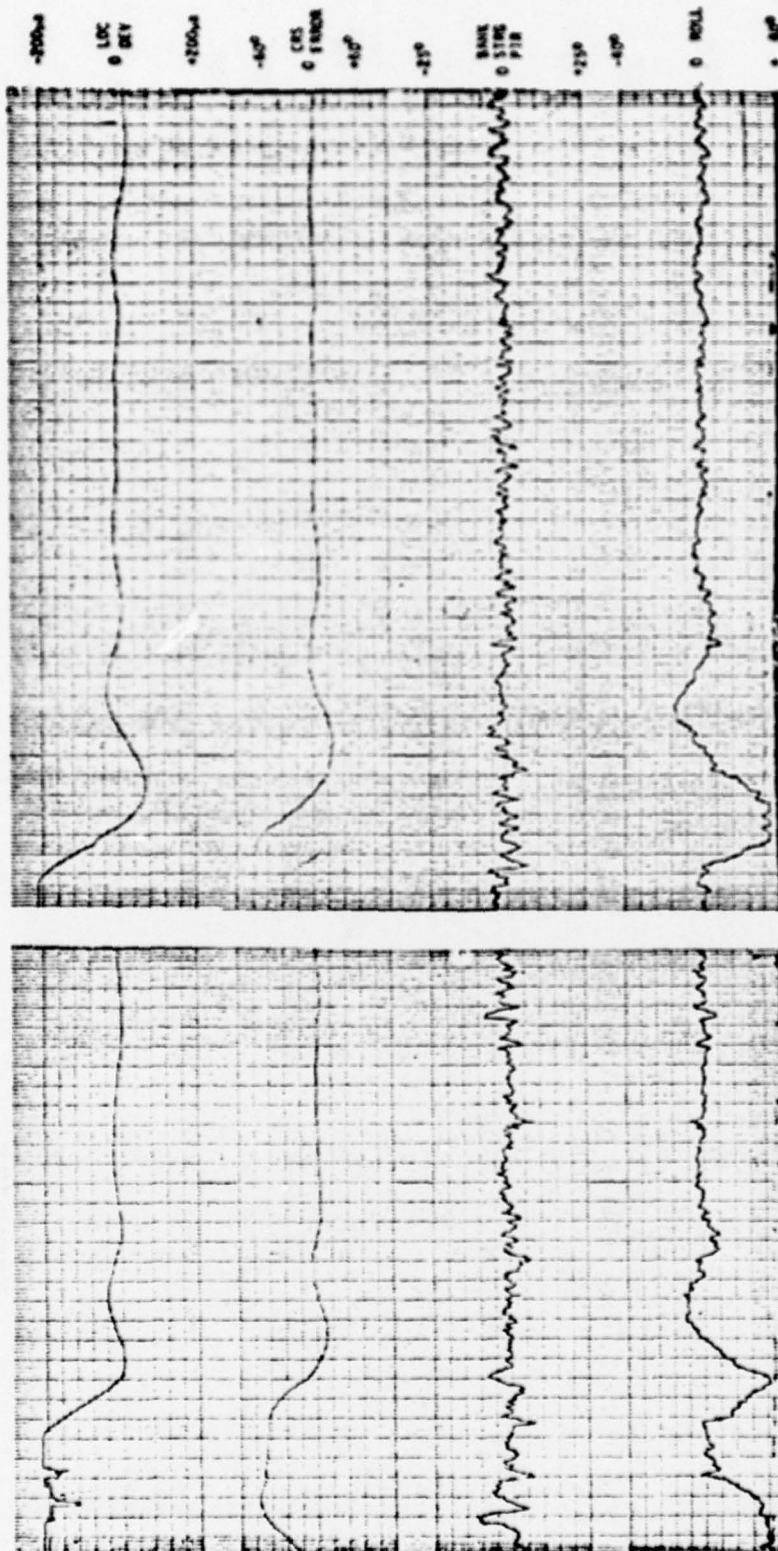
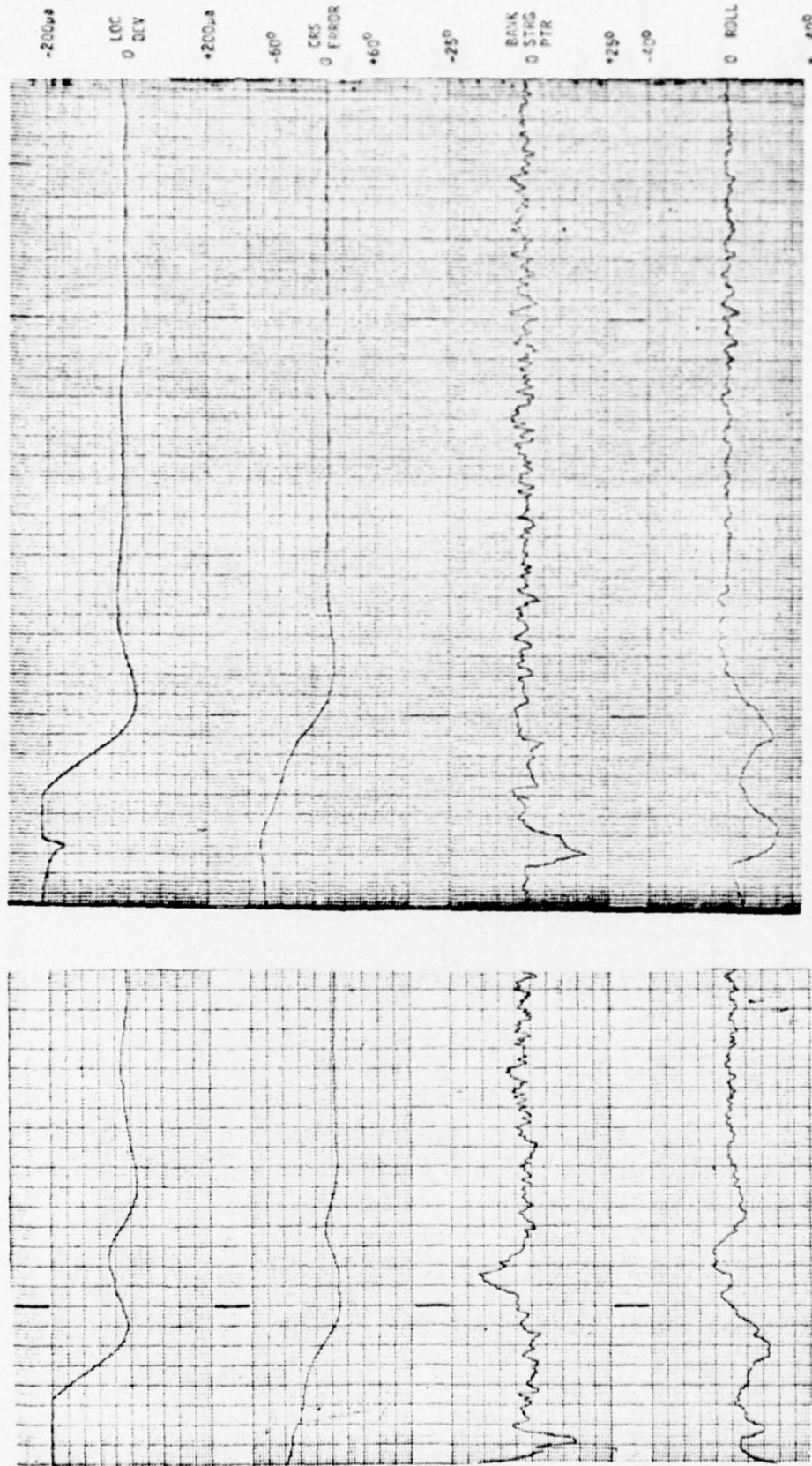


Figure 105. Flight Test Data - ILS

Date: 3/9/75



Airspeed: 220Kts
Intercept Angle: 30°
Range to Ray at Intercept: 8mi

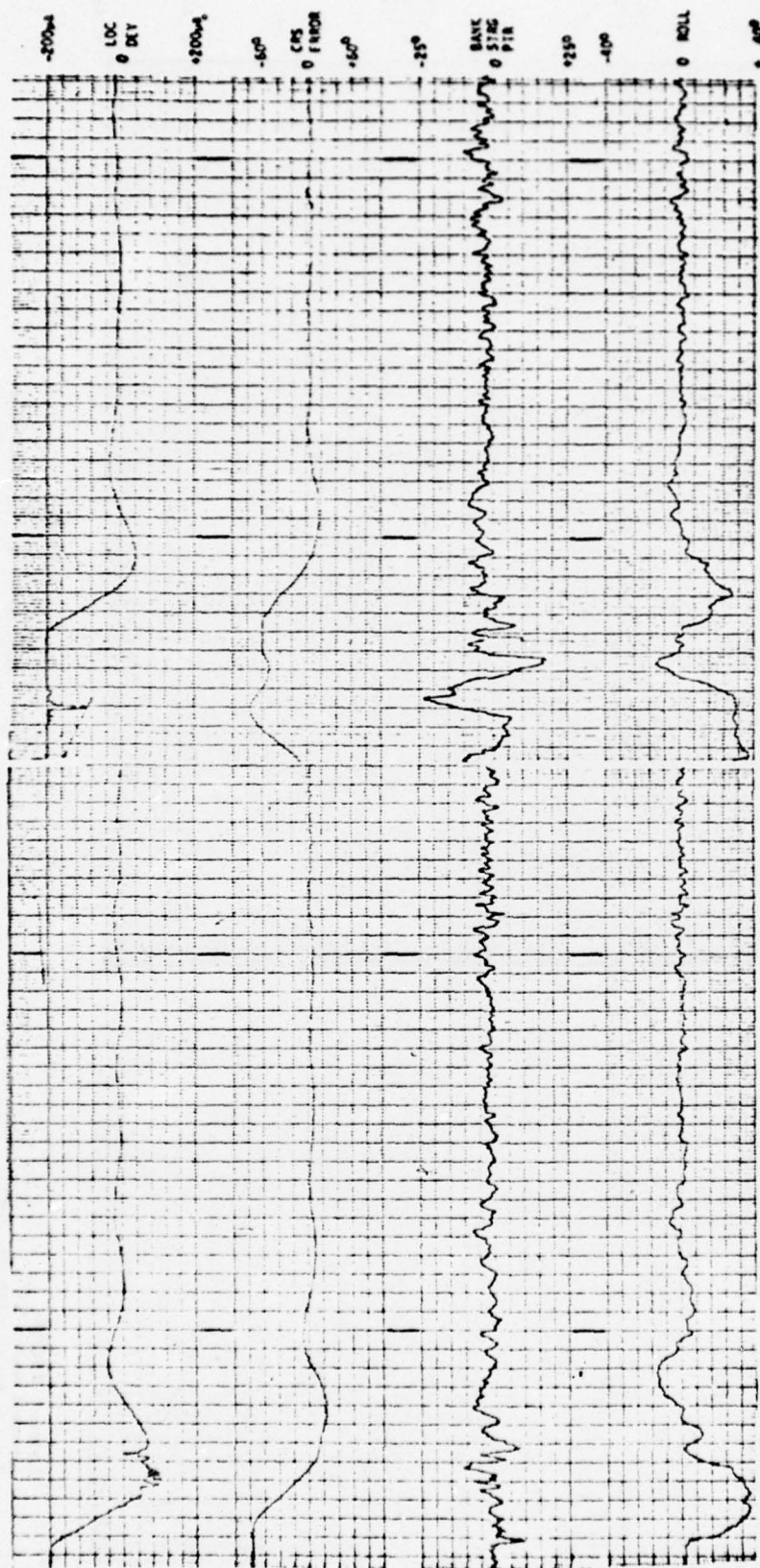
Airspeed: 220Kts
Intercept Angle: 15°
Range to Ray at Intercept: 15mi

Date: 3/10/75

A3

A2

Figure 106. Flight Test Data - ILS



Altitude: 220 Kt
Intercept Angle: 90°
Range to Ray at Intercept: 18 mi

Altitude: 220 Kt
Intercept Angle: 60°
Range to Ray at Intercept: 10 mi

Date: 3/14/75

Figure 107. Flight Test Data - ILS

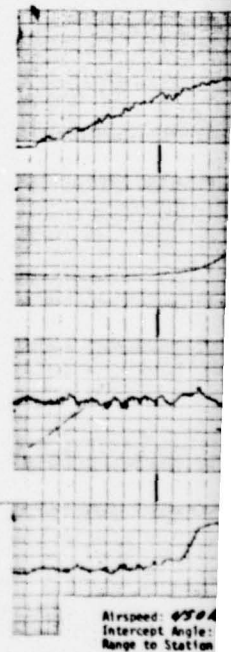
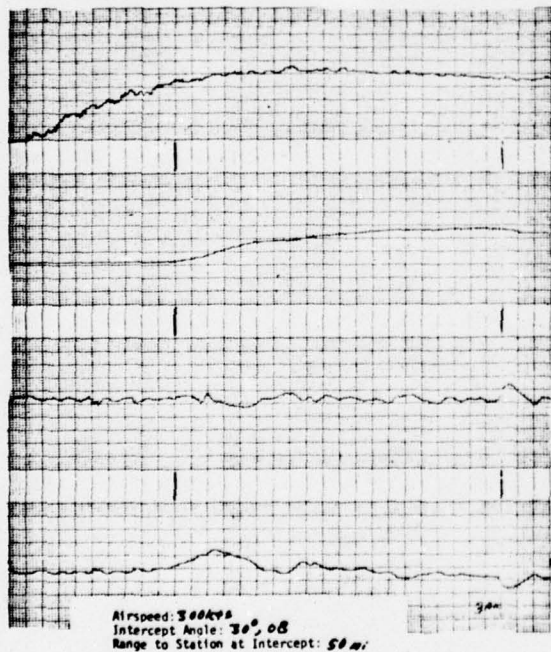
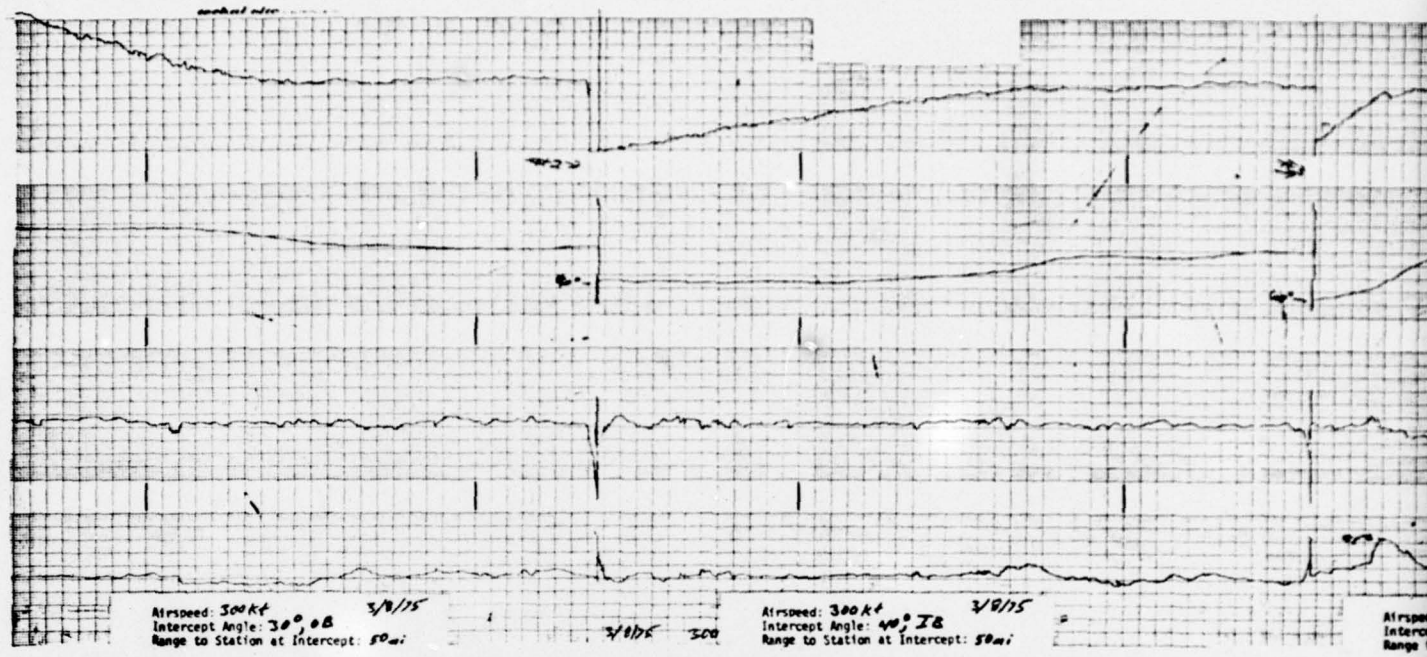


Figure 108. Flight Test Data

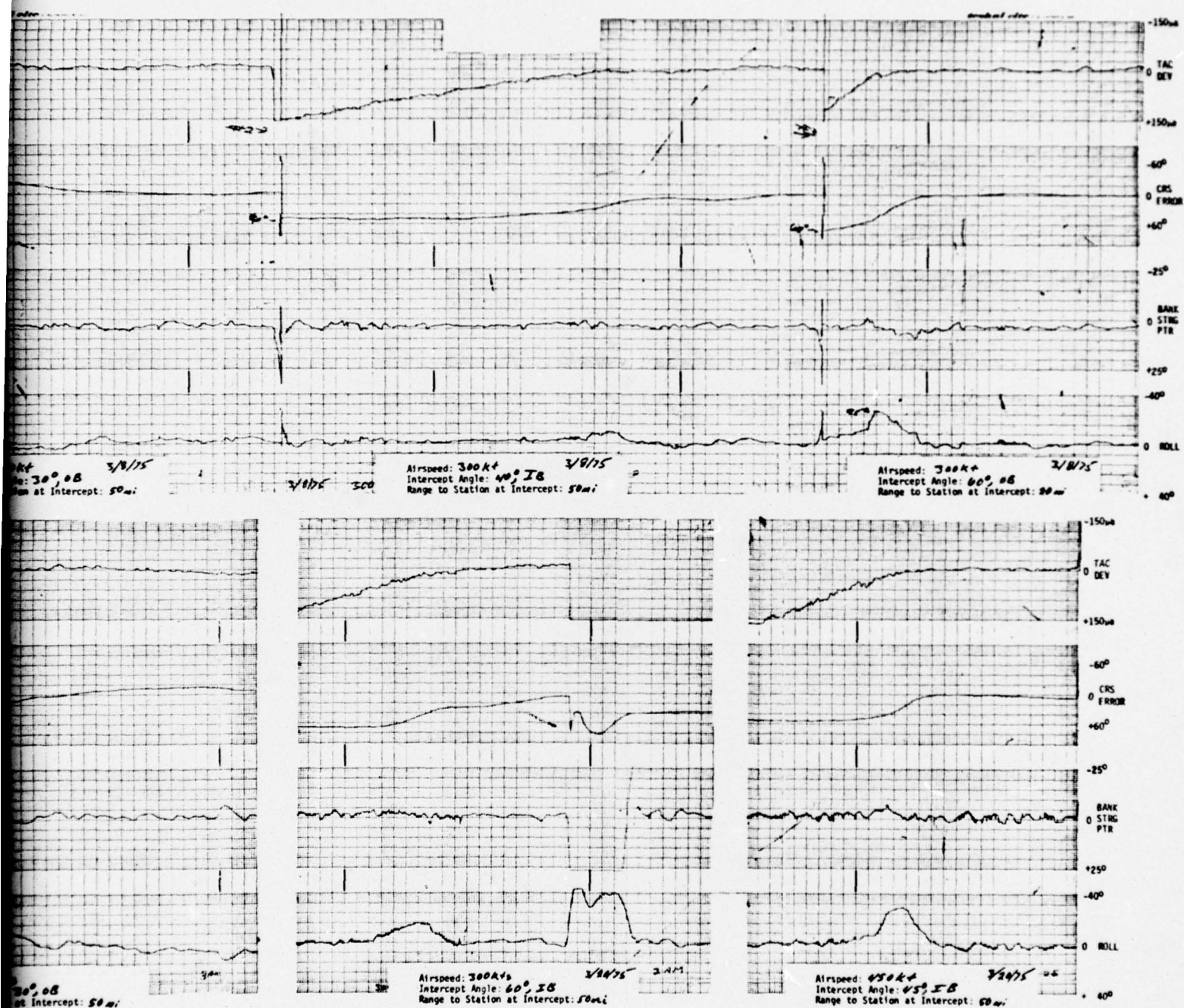


Figure 108. Flight Test Data - TACAN

TABLE 10. DFDC PERFORMANCE RESULTS

March 3 - 21, 1975

TACAN	DISTANCE TO STATION AT BEAM INTERCEPT	INITIAL COURSE ERROR	AIRSPEED	COMMANDED BANK	OVERSHOOT	MAX. TRACK ERROR
1)	50nm	30°	300kt	9°	0	15µa
2)	50nm	40°	300kt	10°	0	10µa
3)	20nm	60°	300kt	25°	0	10µa
4)	50nm	30°	300kt	10°	0	5µa
6)	50nm	60°	300kt	16°	0	10µa
7)	50nm	45°	450kt	30°	0	5µa
<u>ILS</u>						
1)	8nm	30°	170kt	8°	10µa	5µa
2)	10nm	45°	170kt	10°	20µa	10µa
3)	15nm	60°	170kt	16°	30µa	5µa
4)	18nm	90°	170kt	16°	50µa	10µa
*5)	8nm	30°	200kt	30°	30µa	40µa
*6)	8nm	45°	200kt	30°	See Figure 7.9	
7)	10nm	60°	200kt	30°	20µa	20µa
8)	18nm	90°	200kt	30°	60µa	15µa
*9)	8nm	30°	220kt	20°	-50µa	20µa
10)	15nm	45°	220kt	20°	20µa	10µa
11)	10nm	60°	220kt	26°	40µa	20µa
12)	18nm	90°	220kt	30°	60µa	20µa

* Performance data is not representative since bank steering pointer was not maintained near zero.

6.1 Deviations From the Ideal

It is the purpose of this section to describe effects on performance of the DFDC during flight tests caused by equipment limitations: the lack of a Central Air Data Computer (CADC), and a radar altimeter of limited range (0 to 1,000 feet).

6.1.1 Airspeed Inputs

The software variable U , used as true airspeed in the DFDC control law programs, is set equal to either the airspeed entered on the Airspeed Controller or the true airspeed output from the CADC and is selected by means of a toggle switch on the controller panel.

U is used for initialization of crosstrack rate in the Kalman filter, for the computation of the circular bank command, for integration in the range estimation, and for gain programming in the manual heading control law.

If a CADC is provided so that true airspeed is sent to the DFDC, then an accurate indication of true airspeed would be available and would result in an optimum performance. If the manual airspeed is selected via the airspeed controller and the manual airspeed is not set to the correct value, then the DFDC will not perform optimally. The degradation in performance will be a function of the error in the manual airspeed setting.

The effect of improper manual airspeed setting on the gain programming for the heading control laws for too high/low a setting would result in too high/low a gain heading error. This would result in too high/low a value of bank attitude for a given heading error.

Error in manual airspeed will result in a proportional amount of error in the initial Kalman filter position rate term. However, as the filter is updated, the rate term will settle to the correct estimate since the wind estimator will compensate for any errors. The resulting effect on performance of the capture submode will be minimal.

Incorrect manual airspeed setting will effect the range estimation algorithm and therefore performance in the track submode. A too high/low airspeed setting will result in a too low/high range estimation. This will increase/decrease the gain programming on the bank command, resulting in increased bank activity/reduced tracking performance.

6.1.2 Radar Altitude

The DFDC was designed to interface with a radar altimeter with a range of 0 to 5,000 feet. The radar altimeter provided with the aircraft used for flight tests had a range of 0 to 1000 feet requiring a modification to the DFDC analog scaling receiver provided for this input.

The radar altimeter provides altitude information utilized to perform a linearization of the glideslope beam. Radar altitude is also used to perform an update on the localizer range estimation once the glideslope beam has been acquired. Effect on localizer tracking due to incorrect altitude information is minimal.

For a 0 to 5000 foot altimeter the glideslope capture and track performance will be optimum. The conversion to the 0 to 1000 foot altimeter resulted in accepting degraded performance in the glideslope mode above 1000 feet. Where the gain on the glideslope control laws will be reduced by 1000 feet/actual altitude. For example, at 2000 feet, the glideslope gain would be reduced by 50%. This reduces the outer loop gains and results in "loose" tracking of the glideslope beam.

6.1.3 Roll Rate Limit

The DFDC was designed to perform on all aircraft from high performance jet fighters to large transports. The roll rate limit, established for all modes at 5 degrees per second in consideration of the flight profile of a transport aircraft, was considered to be acceptable for a jet fighter under instrument flight conditions. During flight tests, pilots commented that roll rate was too low and the following roll rate limits were recommended:

ILS Approach - 2.5 degrees per second
TACAN, ILS Capture Modes - 7.5 degrees per second
Manual Heading - 10 degrees per second

In the approach mode, 2.5 degrees per second will not support the improved performance provided by the DFDC. The following would be required to modify the roll rate limits in capture and manual heading modes:

- 1) The addition of two constants in DFDC memory.
- 2) The addition of a subroutine to replace the present rate limit instruction, capable of determining mode and selecting the correct constant.
- 3) Verification of performance with an external memory and the Collins Hybrid Simulation Facilities.
- 4) The replacement of approximately eight programable read-only memories.

APPENDIX A

SYSTEM FLOW CHARTS

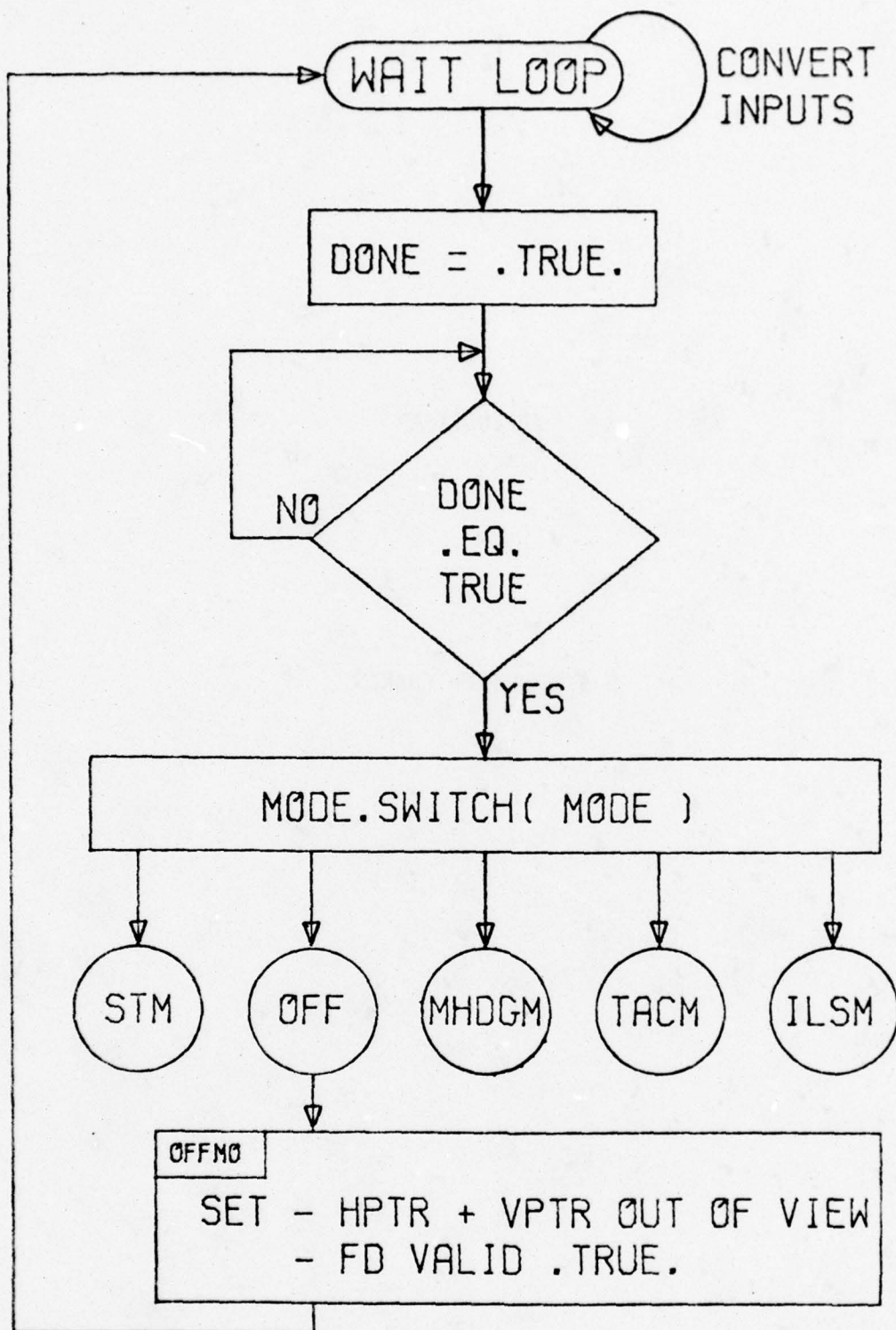


Figure 109. Wait Loop and Off Mode Control

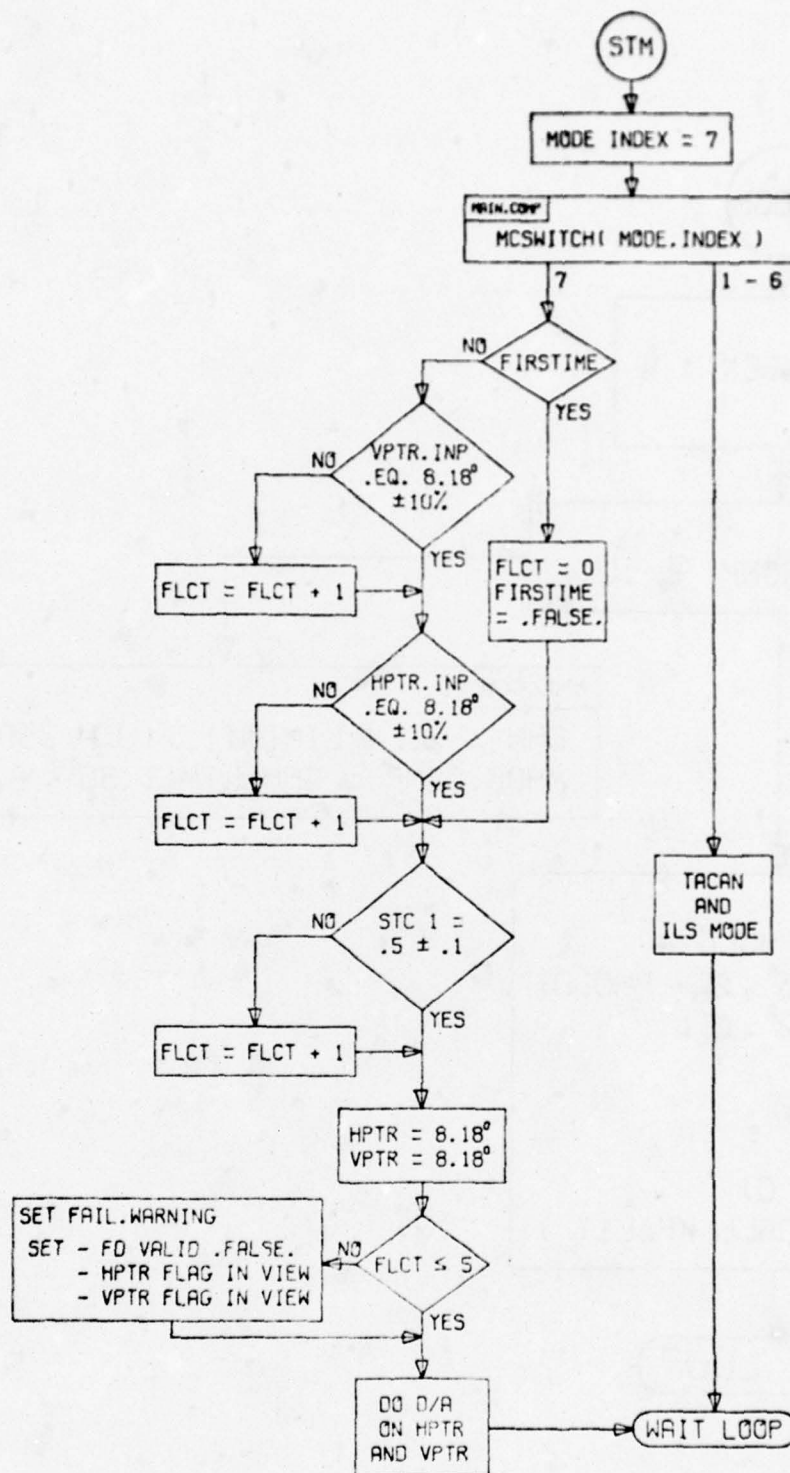


Figure 110. STM Mode Control

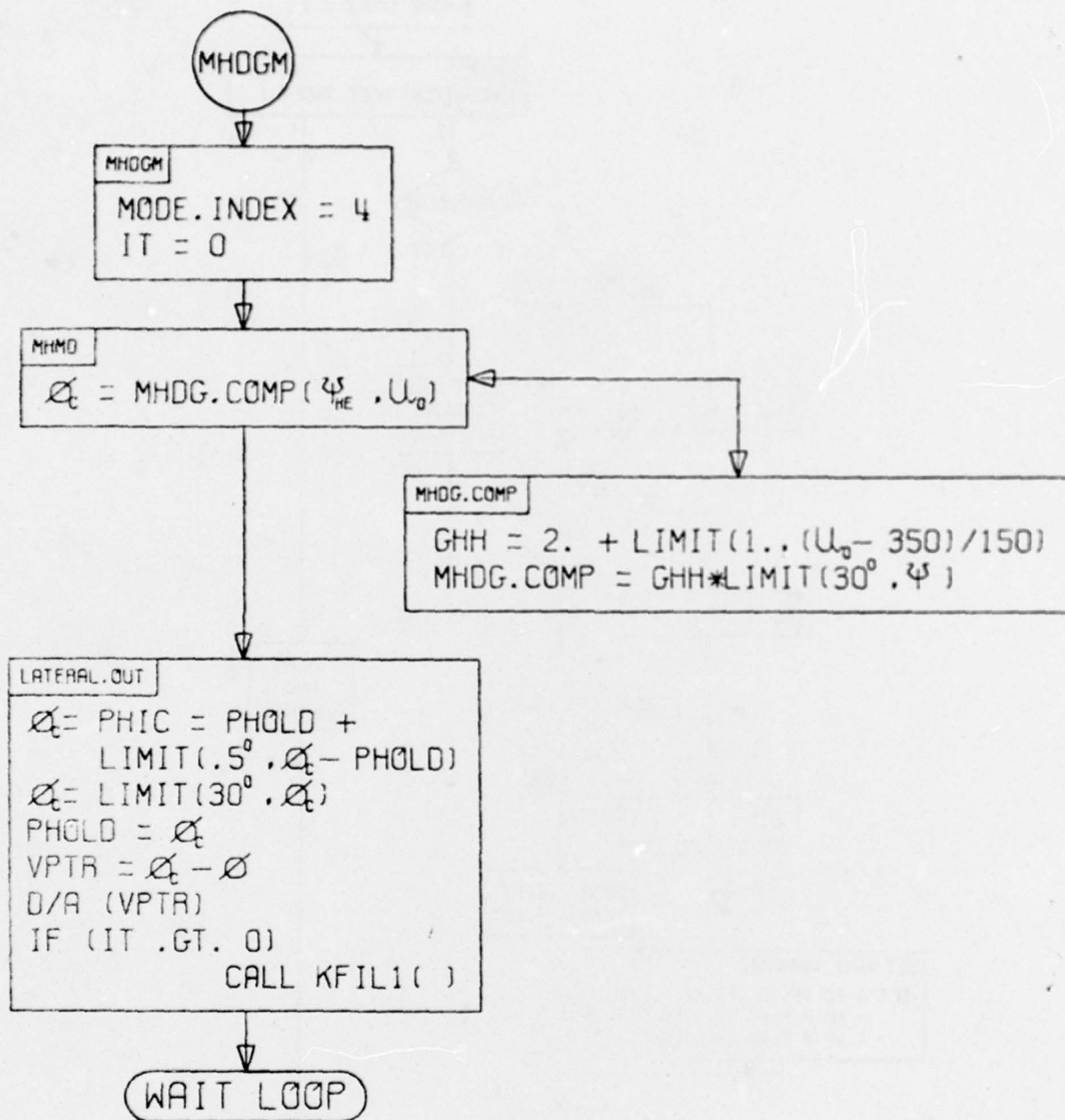


Figure 111. Manual Heading Mode Control

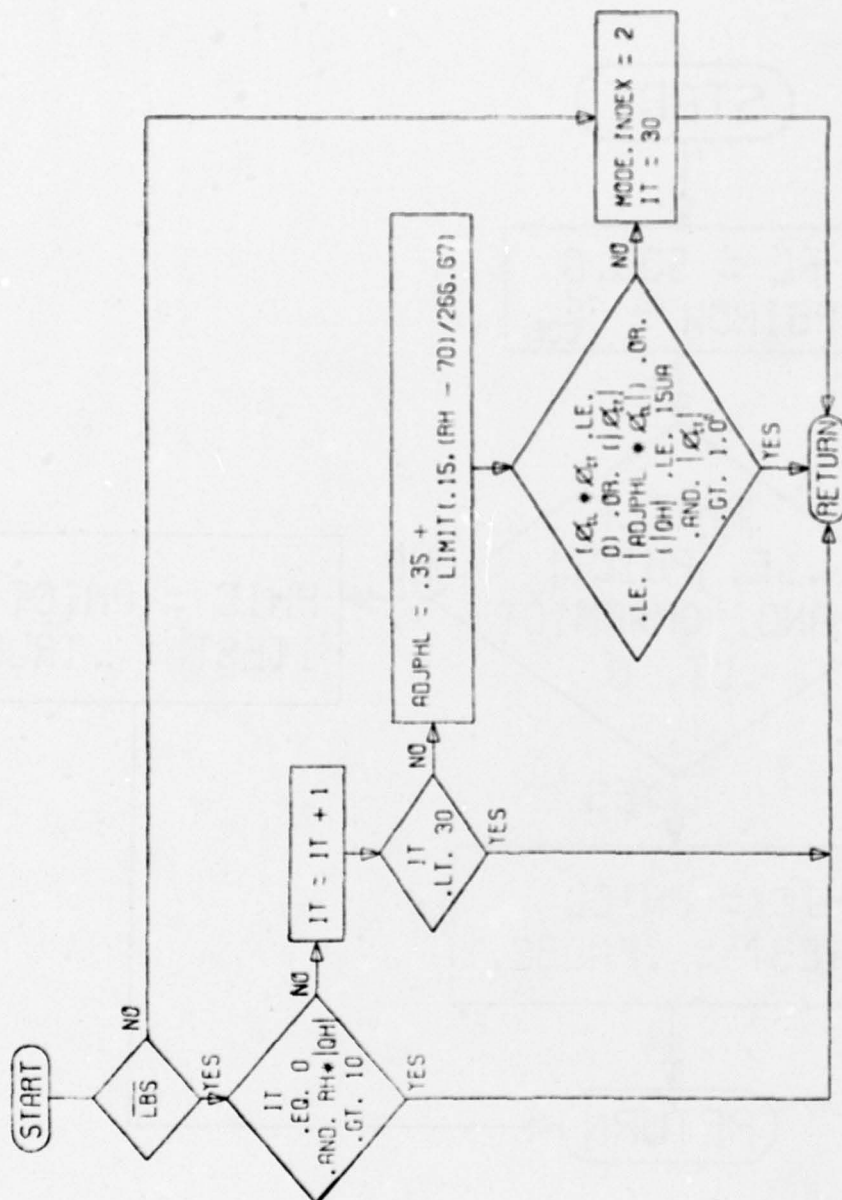


Figure 112. TACAN Manual Heading to Capture

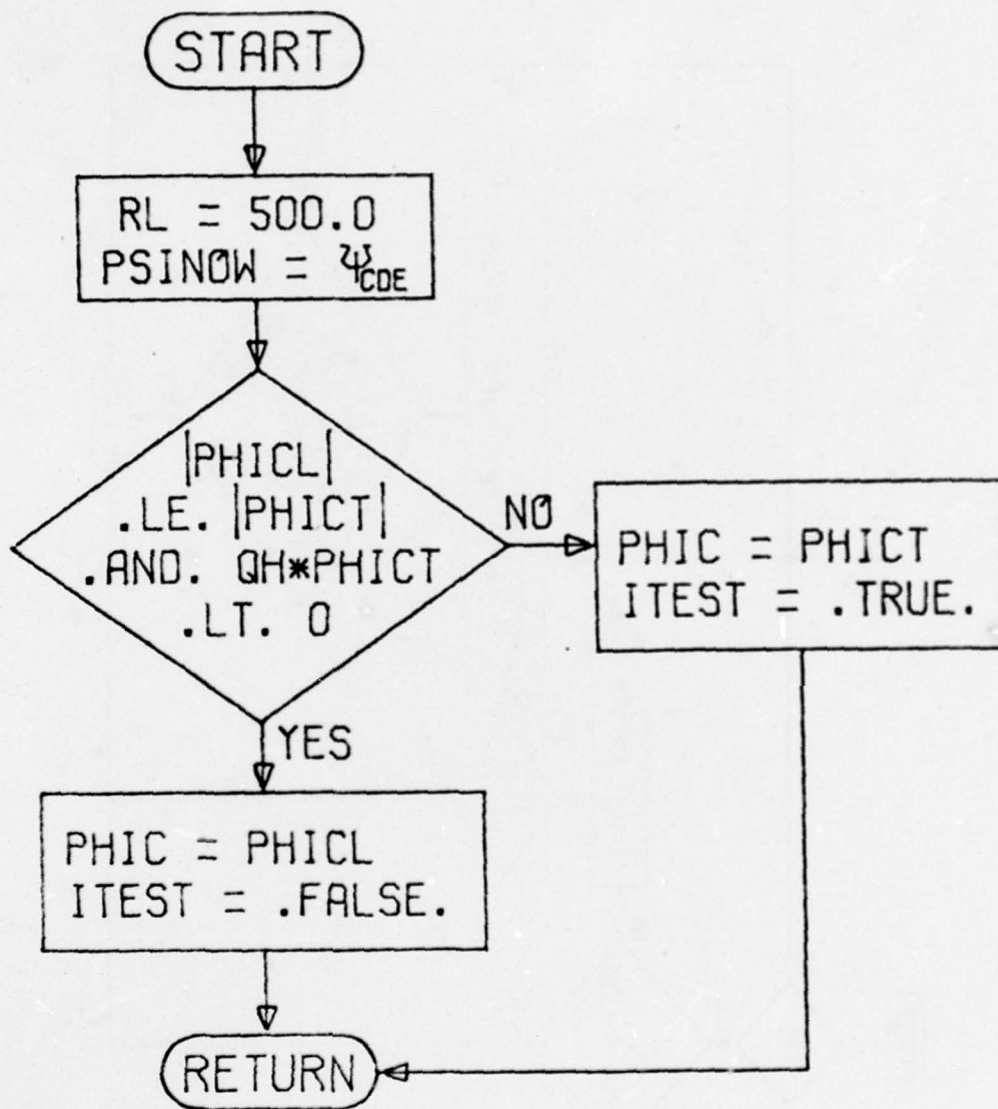


Figure 113. TACAN Capture Computation

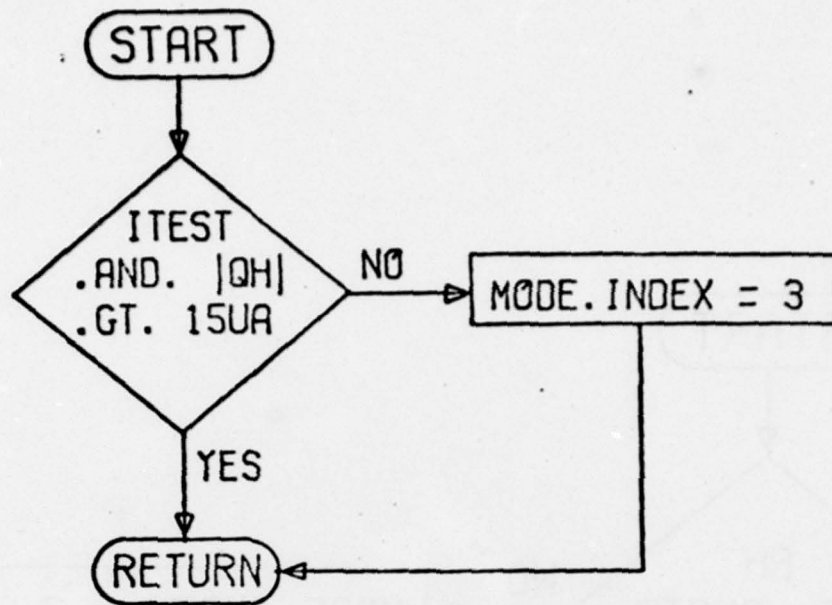


Figure 114. TACAN Capture to Track

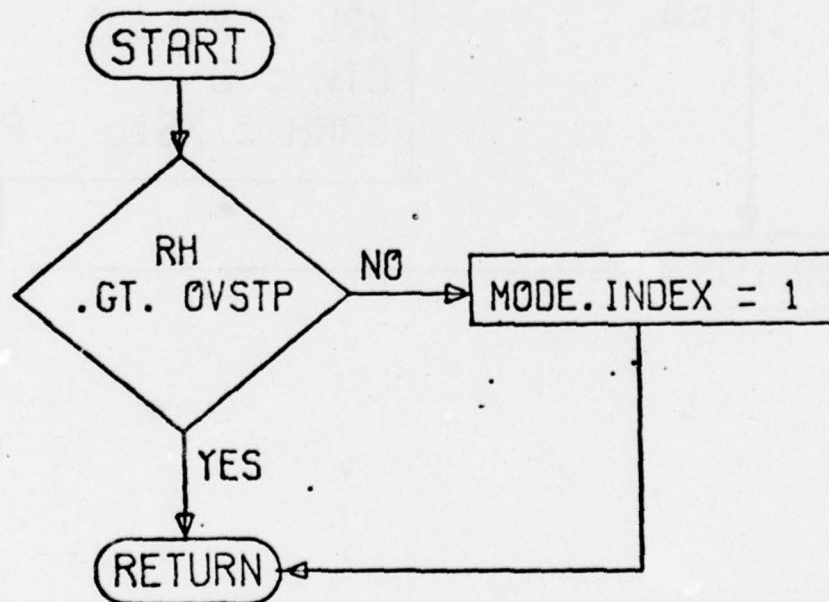


Figure 115. TACAN Track to Overstation

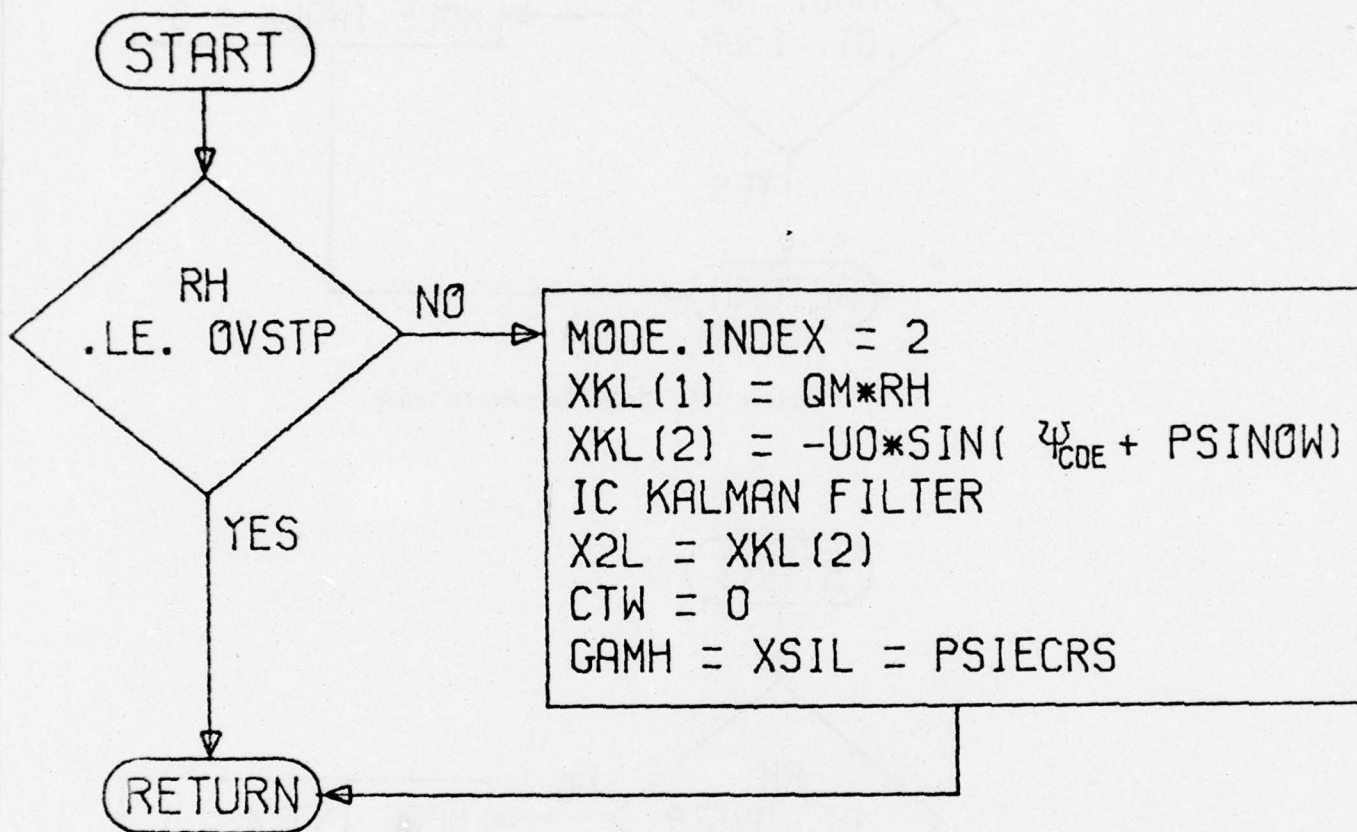


Figure 116. TACAN Overstation to Capture

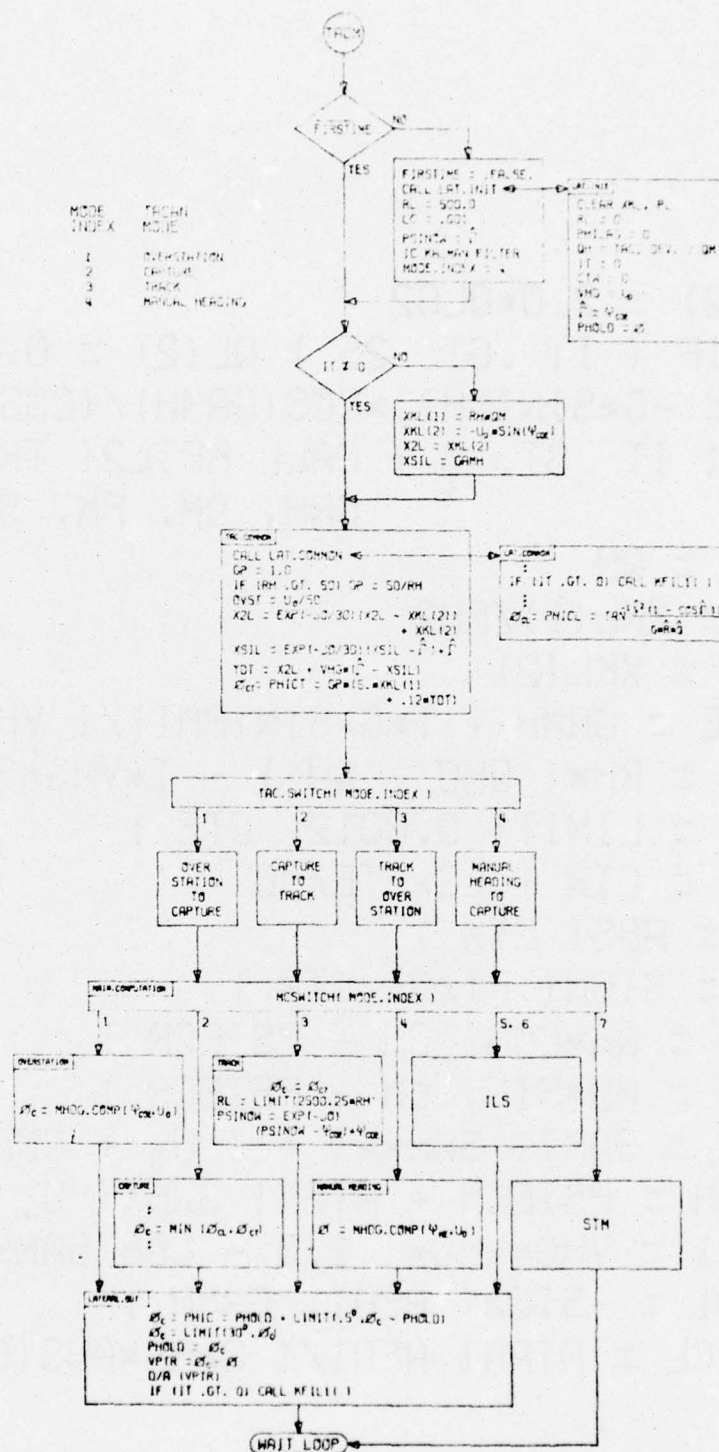


Figure 117. TACAN Mode Control

```

QL(2) = 2.0*QL02
  IF ( IT .GT. 25 ) QL(2) = 0.005*QL02
FK = -G*SIN(PHI)*COS(GAMH)/(COS(PHI)*RH1)
  IF ( IT .GT. 0 ) CALL KFIL2( TRLAT, XKL,
                                DRH, QM, FK, SDUM, CDUM )

QHO = QH
QH = XKL(1)/RH2
QDB = XKL(2)
GAME = GAMH + T*G*SIN(PHI)/(VHG*COS(PHI) )
CTE = RH*( QHO - QH ) - T*VHG*SIN(GAME)
CTE = LIMIT( 0.0012, CTE )
CTW = CTW + LC*CTE/JO
WH = ABS( CTW )
OH = SIGN( PI/2, CTW )
CWH = WH*COS( OH - PSIECR )
SWH = WH*SIN( OH - PSIECR )
VHG = SQRT( SWH**2 + ( U0 + CWH)**2 )
GAMH = PSIECR + ATAN( SWH/( U0 + CWH ) )
AED1 = VHG*VHG*( 1.0 - COS(GAMH) )
AED1 = -SIGN( AED1, GAMH )
PHICL = ATAN( AED1/( G*RH*ABS(QH) ) )

```

Figure 118. Lat. Common

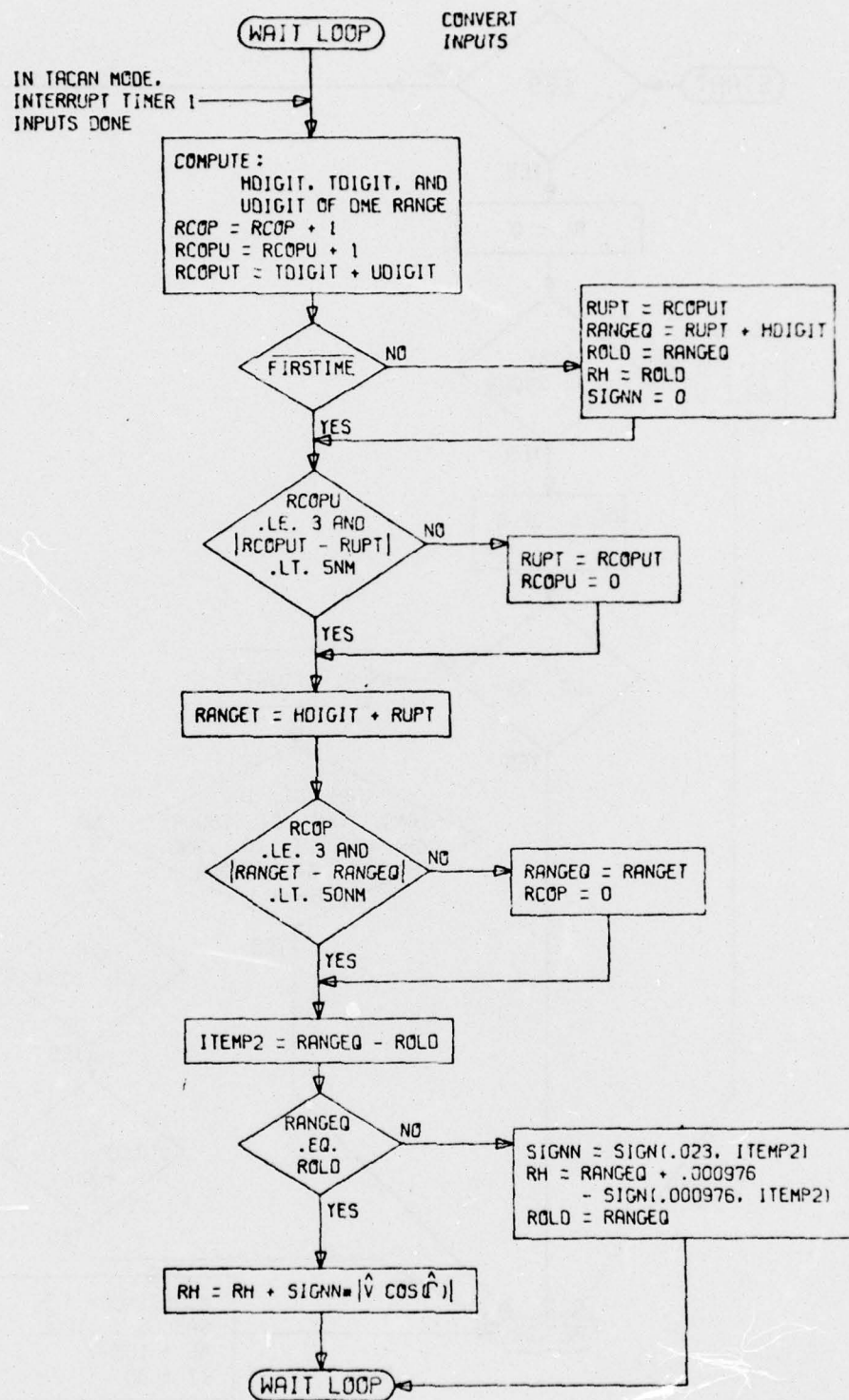


Figure 119. DME Range Smoothing Algorithm

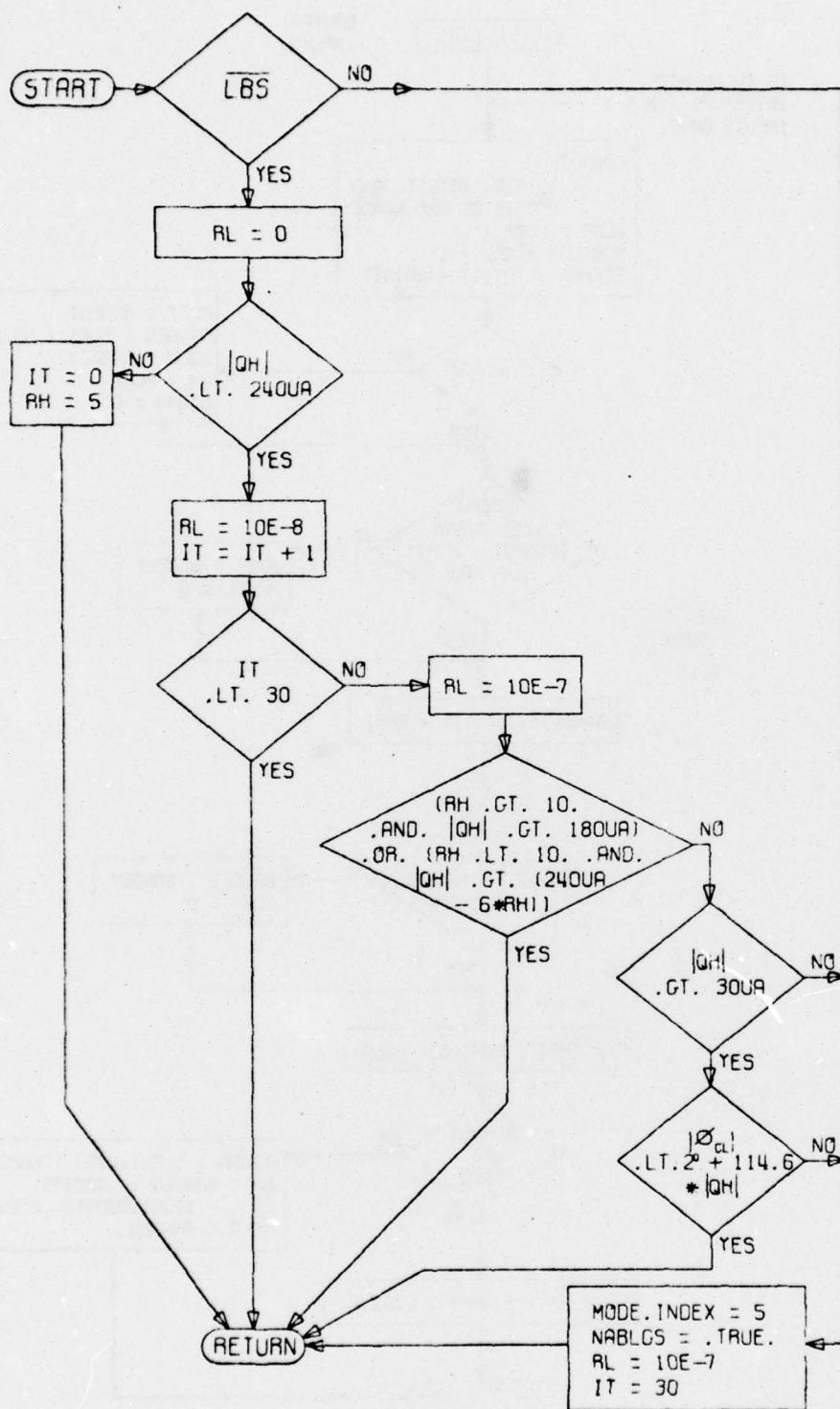


Figure 120. ILS Localizer Manual Heading to Capture

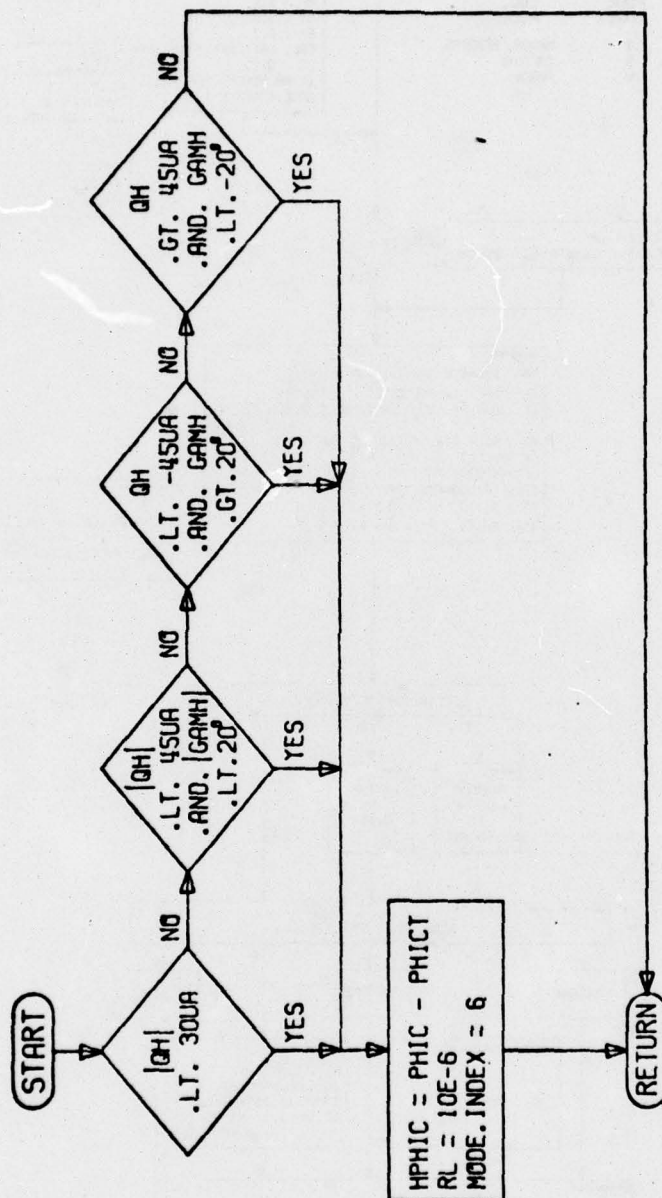


Figure 121. ILS Localizer Capture to Track

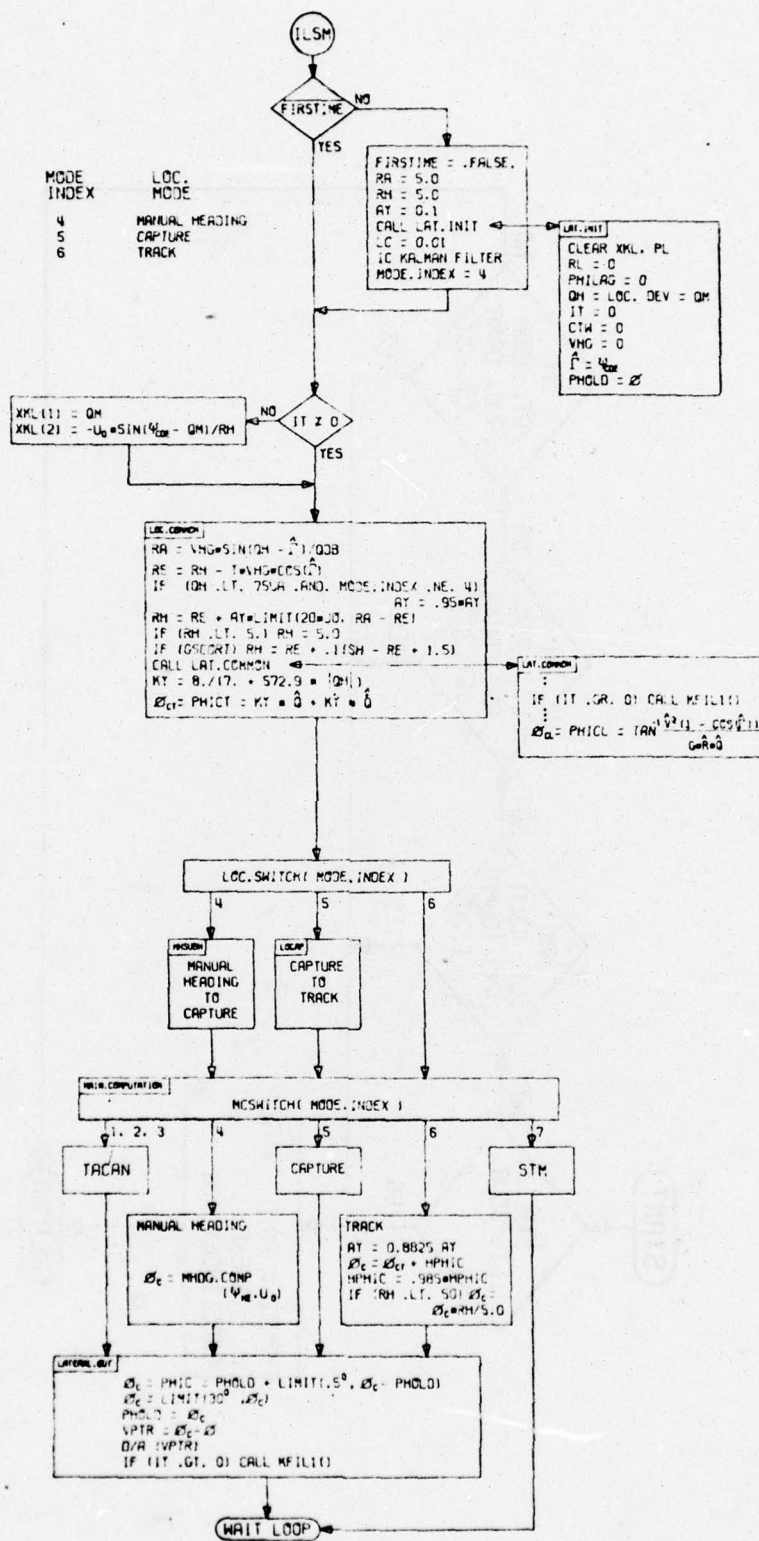


Figure 122. ILS Localizer Mode Control

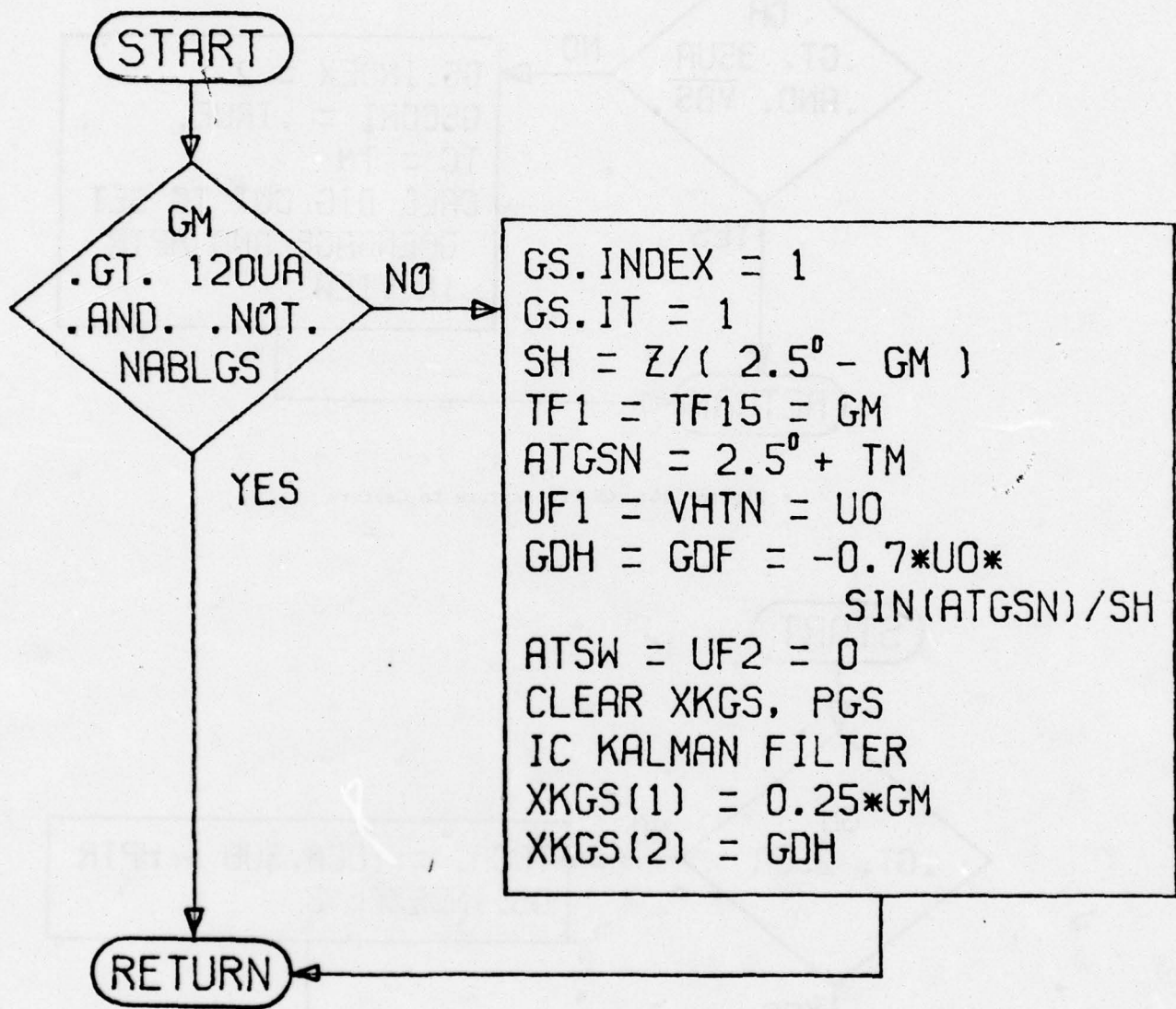


Figure 123. GS Idle to Pre-capture

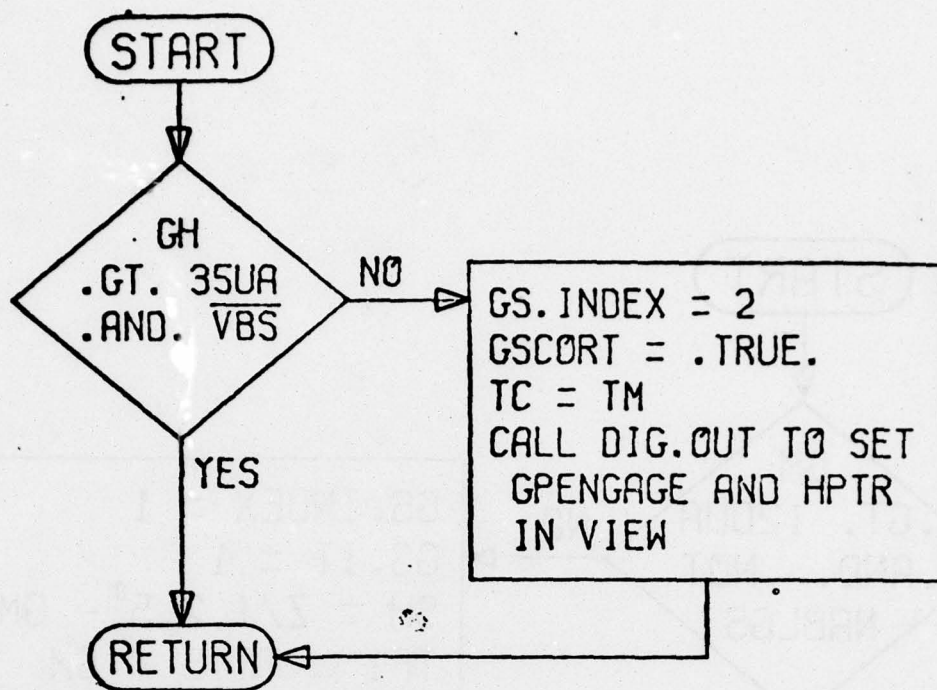


Figure 124. GS Pre-capture to Capture

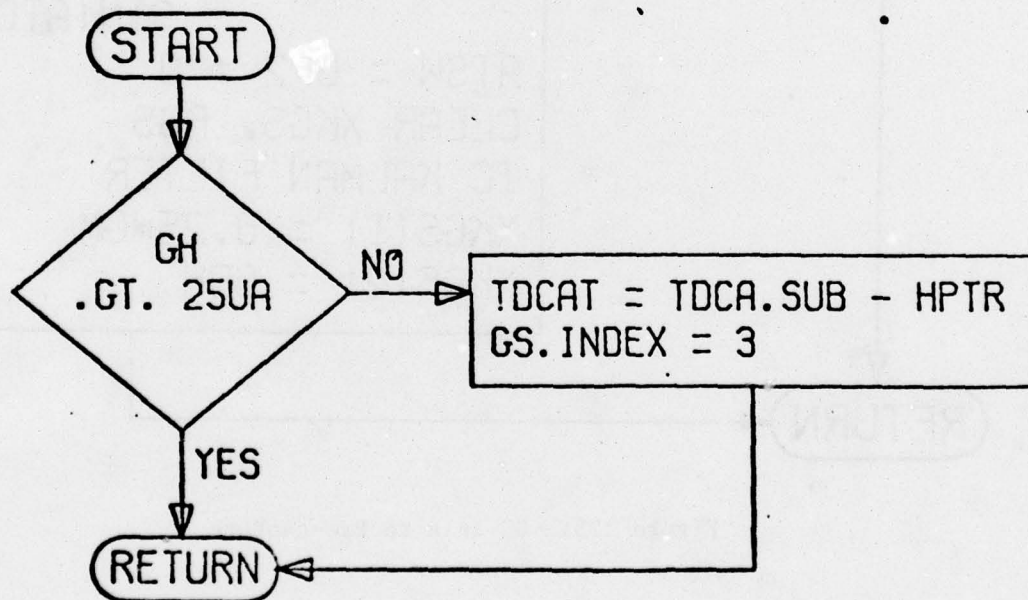


Figure 125. GS Capture to Track

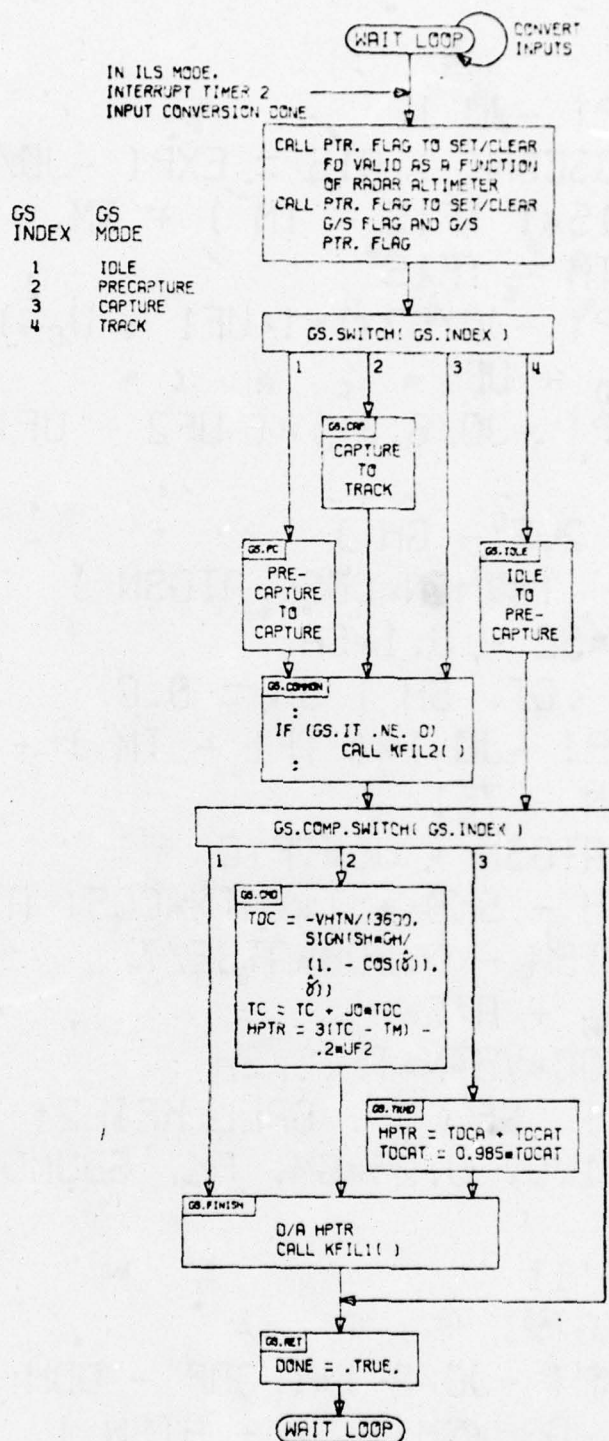


Figure 126. Glideslope Mode Control

```

A15 = EXP( -J0 )
  IF ( GSCORT ) A15 = EXP( -J0/15 )
TF15 = A15*( TF15 - TM ) + TM
TF15C = TM - TF15
UF1 = EXP( -J0/63 )*( UF1 - U0 ) + U0
UF1C = U0 - UF1
UF2 = EXP( -J0/6.3 )*( UF2 - UF1C ) + UF1C
SH0 = SH
SA = Z/( 2.50 - GH )
SE = SH - T*VHTN*COS( ATGSN )
SH = 0.9*SE + 0.1*SA
IF ( 0.0 .GT. SH ) SH = 0.0
TF1 = EXP( -J0 )*( TF1 - TM ) + TM
TF1C = TM - TF1
ATGSE = ATGSN + J0*TF1C
ATSE = SH - SH0 + T*VHTN*COS( ATGSE )
ATSW = ATSW - 0.008*ATSE/T
VHTN = U0 + ATSW
FK = -3600*VHTN*TF1C/SH
IF ( GSIT .NE. 0 ) CALL KFIL2( TRGS, XKGS,
    1.0, 0.25*GM, FK, SDUMG, CDUMG )
GSIT = 1
GH = XKG(1)
GDH = XKG(2)
GDF = EXP( -J0/2 )*( GDF - GDH ) + GDH
ATGSN = -GDH*SH/( U0 + ATSW )

```

Figure 127. Glideslope Common

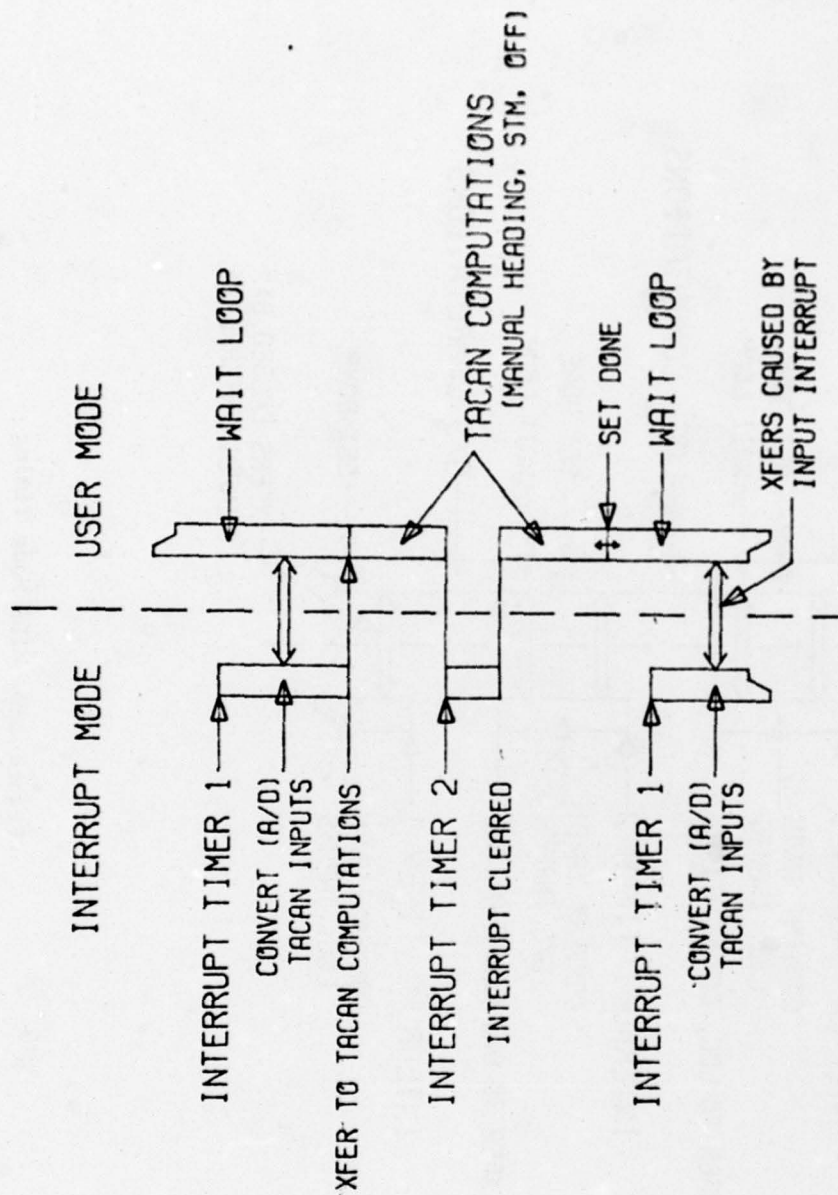


Figure 128. TACAN Mode Timing (All Models Except ILS)

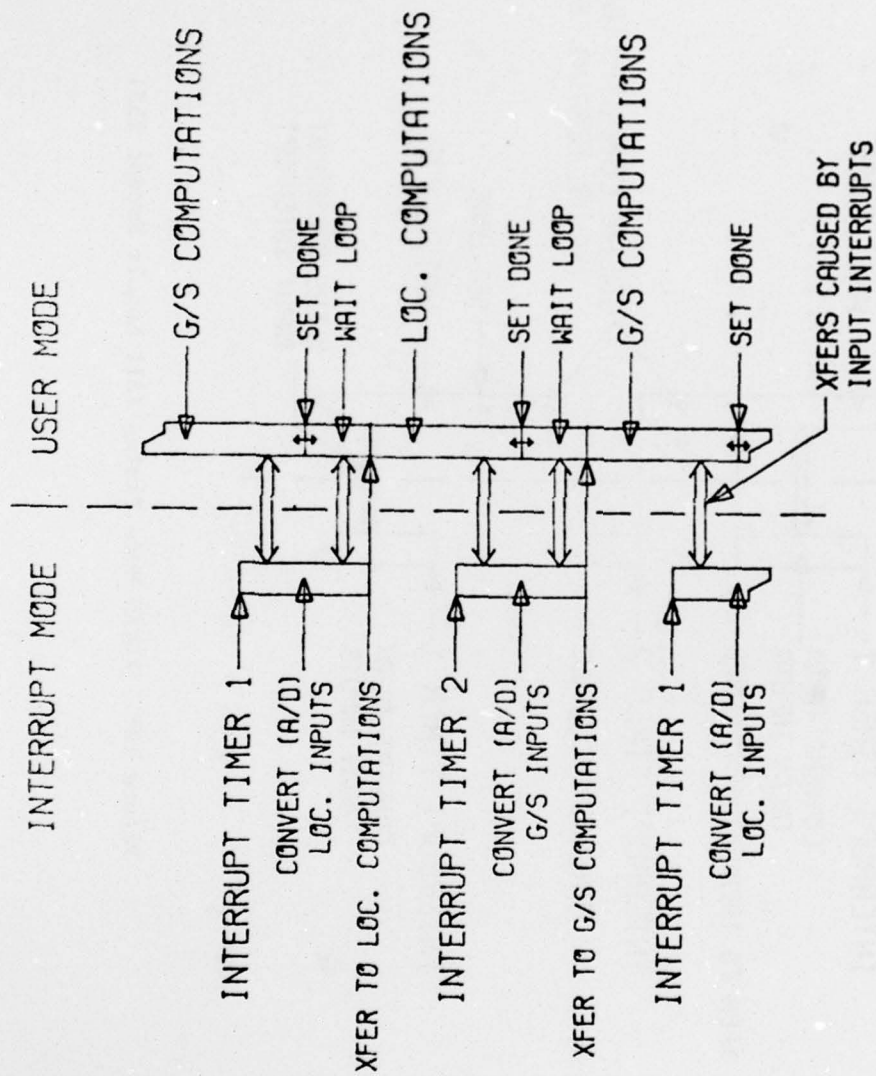


Figure 129. ILS Mode Timing

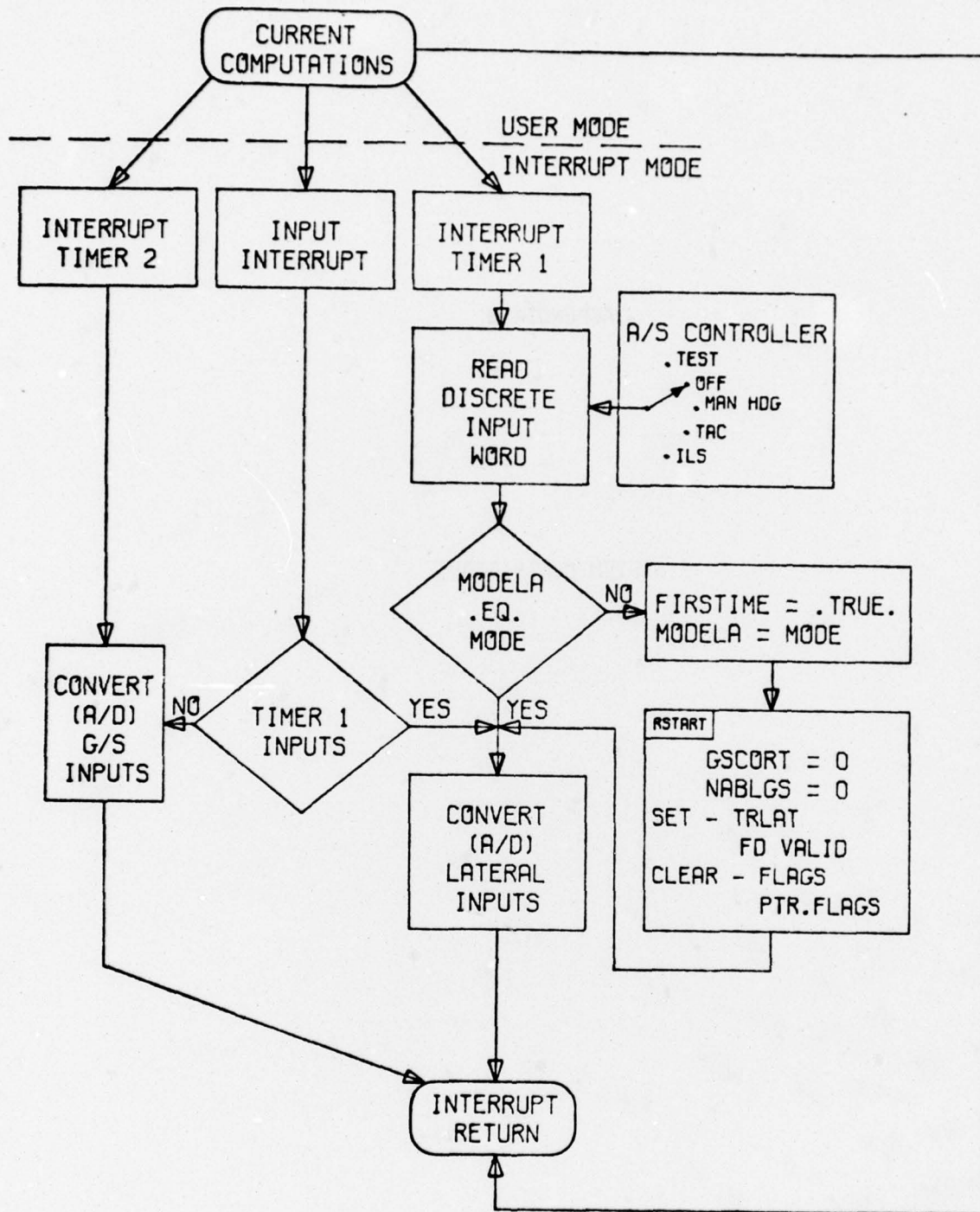


Figure 130. Mode Control

APPENDIX B

SYSTEM DEFINITIONS

THIS APPENDIX CONTAINS THE DEFINITION OF VARIABLES CODED IN THE SOFTWARE PROGRAMS AND ILLUSTRATED IN THE FLOW CHARTS.

DEFINITIONS

GENERAL TERMS

AED1	DUMMY VARIABLE
AED2	DUMMY VARIABLE
AED3	DUMMY VARIABLE
DTR	DEGREES TO RADIAN'S CONVERSION CONSTANT
G	GRAVITY
GSA	G/S BEAM ANGLE
JO	SAMPLE TIME, SECONDS
T	SAMPLE TIME, HOURS

BASIC AIRCRAFT TERMS

GAM	LATERAL FLIGHT PATH ANGLE
GGAM	G/S FLIGHT PATH ANGLE
GM	G/S BEAM DEVIATION, MEASURED
GNF	G/S BEAM DEVIATION, NOISE FREE
HM	HEADING ANGLE TO BE FLOWN
PSI	
PSIECR	COURSE DATUM ERROR
PSIECRS	
PSIEMH	
PSIEMHDG	MANUAL HEADING ERROR
PHI	BANK ANGLE
QM	LATERAL BEAM DEVIATION, MEASURED
QNF	LATERAL BEAM DEVIATION, NOISE FREE
RANGE	RANGE, RADIAL
TOM	PITCH RATE, MEASURED
TM	PITCH ANGLE, MEASURED

U0	AIRSPEED
X	ALONG TRACK DISTANCE TO THRESHOLD
XD	ALONG TRACK RATE
Y	CROSS TRACK DISTANCE
YD	CROSS TRACK RATE
Z	ALTITUDE

LATERAL AXIS TERMS

A15	DUMMY VARIABLE
ATE	ALONG TRACK ERROR (WIND COMPUTATION)
ATW	ALONG TRACK WIND
AY	GAIN ADJUSTMENT USED IN THE PREDICTOR/CORRELATOR \dot{R} FILTER
CTW	CROSS TRACK WIND
CTE	CROSS TRACK ERROR (WIND COMPUTATION)
CWH	COSINE OF WIND (INTERMEDIATE QUANTITY)
DQH	DUMMY VARIABLE FOR PASSING RH
DQM	DUMMY VARIABLE FOR PASSING QM
FTIME	
FIRSTIME	FIRST TIME
GAME	GAMMA, PREDICTED, , USED IN WIND MEASUREMENT
GAMH	GAMMA, BEST ESTIMATE, , ANGLE TO VELOCITY VECTOR
GHH	GAIN PROGRAMMING TERM FOR HEADIN HOLD MODE
IT	ITERATION TIME, USED FOR CAPTURE DETERMINATION
KCTE	SCALING TERM FOR CTE
KSCAL1	SCALING CONSTANT 1
KSCAL2	SCALING CONSTANT 2
LATSW	LATERAL AXIS SWITCH
LC	CROSS-TRACK WIND FILTER COEFFICIENT
MODE	MODE, DESIRED - STM, OFF, MAN HDG, TAC, ILS

MODELA	MODE LATERAL AXIS
OH	WIND ANGLE, BEST ESTIMATE
PHIC	BANK ANGLE COMMAND
PHICL	PHIC CAPTURE
PHICT	PHIC TRACK
PHII	DIFFERENCE BETWEEN PRESENT AND PREVIOUS BANK COMMANDS
PHILAG	PHI FILTERED
PHOLD	LAST VALUE OF PHIC, USED IN DETERMINING ROLL RATE
PL	COVARIANCE MATRIX
QDB	BEAM RATE, BEST ESTIMATE
QH	BEAM DEVIATION, BEST ESTIMATE
QH0	PREVIOUS VALUE OF QH
QL	DISTURBANCE VECTOR, NOISE
QL02	QL(2) FIXED REFERENCE VALUE
RH	RANGE, BEST ESTIMATE
RL	SIGNAL/NOISE RATIO TERM
RN	DESIRED (COMPUTED) RADIUS OF CURVATURE TO BE FLOWN
SQM	DUMMY VARIABLE FOR PASSING QM
SWH	SINE OF WIND (INTERMEDIATE QUANTITY)
TRLAT	TRANSITION MATRIX
VHG	GROUND VELOCITY, BEST ESTIMATE
WH	WIND MAGNITUDE, BEST ESTIMATE
XMANHO	SUBROUTINE TO COMPUTE MANUAL HEADING BANK COMMAND
TACAN TERMS	
ADJPHL	SCALING TERM ON PHICL TO TRIP TO TACAN CAPTURE
DME R	DME RANGE
GP	GAIN PROGRAMMING TERM FOR PHICT
OVSTP	OVER STATION TRIP POINT

PSINOW PSI FILTERED
 QM
 TGM TACAN BEAM DEVIATION
 X2L \dot{Y} , FILTERED XKL(2)
 XKL(1) CROSS TRACK DISTANCE, BEST ESTIMATE
 XKL(2) CROSS TRACK RATE, BEST ESTIMATE
 XKL(3) BEAM NOISE, BEST ESTIMATE
 XKL(4) GUST NOISE, BEST ESTIMATE
 XS1L GAMH, FILTERED
 YDT CROSS TRACK RATE, \dot{Y}

ILS (LOC) TERMS

AY RANGE FILTER COEFFICIENT
 HPHIC PHI COMMAND DIFFERENCE TO BE WASHED OUT IN TRANSITION
 FROM CAPTURE TO TRACK
 KY SCALING TERM FOR Y FED INTO PHICT
 NABLS ENABLE G/S
 RA MEASURED RANGE DERIVED FROM QDB
 RD RANGE DIFFERENCE BETWEEN RA AND RE
 RE RANGE, PREDICTED
 RH0 PREVIOUS BEST ESTIMATE OF RH
 XKL(1) BEAM POSITION, BEST ESTIMATE
 XKL(2) BEAM RATE, BEST ESTIMATE
 XKL(3) BEAM NOISE, BEST ESTIMATE
 XKL(4) GUST NOISE, BEST ESTIMATE

LONGITUDINAL AXIS (G/S) TERMS

ATGSE ANGLE TO G/S, PREDICTED
 ATGSN ANGLE TO G/S, BEST ESTIMATE
 ATSE ALONG TRACK SLANT WIND ERROR
 ATSW ALONG TRACK SLANT WIND
 CDUMG DUMMY COLUMN VECTOR
 GDF BEAM RATE, FILTERED
 GDH BEAM RATE, BEST ESTIMATE

GH	BEAM DEVIATION, BEST ESTIMATE
GSCORT	G/S CAPTURE .OR. TRACK
GSIT	G/S ITERATION TIME
GSSW	G/S SWITCH
KATSW	SCALING CONSTANT FOR ATSW
PG	
PGS	COVARIANCE MATRIX
QG	
QGS	DISTURBANCE VECTOR, NOISE
RG	S/N RATIO TERM
RPN	RADIUS OF CURVATURE FOR G/S PITCH CAPTURE COMMAND
SA	SLANT RANGE, MEASURED
SE	SLANT RANGE, PREDICTED
SH	SLANT RANGE, BEST ESTIMATE
SHI	SLANT RANGE, IDEAL
SDUMG	SCALED DUMMY VARIABLE
TDC	PITCH RATE COMMAND, CAPTURE
TDCA	PITCH COMMAND, TRACK
TDCAT	PITCH COMMAND DIFFERENCE TO BE WASHED OUT IN TRANSITION FROM CAPTURE TO TRACK
TC	PITCH COMMAND, CAPTURE
TF1	PITCH, FILTERED
TF1C	PITCH RATE
TF1S	PITCH, FILTERED
TF1SC	PITCH RATE
TRANGS	
TRGS	TRANSITION MATRIX
UF1	U0, FILTED
UF2C	U0 RATE
UF2	U0 ACCELERATION

VHTN	VELOCITY TANGENT TO RPN, BEST ESTIMATE
XKG	
XKGS	POSITION VECTOR, BEST ESTIMATE
XKGS(1)	BEAM POSITION, BEST ESTIMATE
XKGS(2)	BEAM RATE, BEST ESTIMATE
XKGS(3)	BEAM NOISE, BEST ESTIMATE
XKGS(4)	GUST NOISE, BEST ESTIMATE

KALMAN FILTER TERMS

CDUM	DUMMY COLUMN VECTOR
FK	ACCELERATION FORCING FUNCTION
P	COVARIANCE MATRIX
PP	COVARIANCE MATRIX
Q	DISTURBANCE VECTOR
R	NOISE COVARIANCE MATRIX, MODELS S/N RATIO
SDUM	SCALED DUMMY VARIABLE
TRANS	TRANSITION MATRIX
XK	POSITION VECTOR, BEST ESTIMATE
ZK	POSITION, MEASURED

HYBRID SIMULATION TERMS

AH	ALT. HOLD VALUE
HF15	ALTITUDE FILTERED
HF15C	ALTITUDE RATE
IATSW	CONTROL TO SET ATSW = 0
ICOUNT	ITERATION COUNT, TOTAL
IGGAM	CONTROL TO SET ATGSN = G/S GAM
IGH	CONTROL TO SET GH = GM
IGDH	CONTROL TO SET GDM = MEASURED BEAM RATE
IQH	CONTROL TO SET QH = QM
IQDB	CONTROL TO SET QDB TO MEASURE RATE VALUE

ISH	CONTROL TO SET SH = SHI
ITAPE	CONTROL FOR WRITING DATA TO TAPE
TMF1	PITCH, FILTERED
TMF1C	PITCH RATE
KFK	SCALING CONSTANT ENTERED VIA KEYBOARD
KKCTW	SCALING CONSTANT ENTERED VIA KEYBOARD
KTDCT	SCALING CONSTANT ENTERED VIA KEYBOARD
KTDCC	SCALING CONSTANT ENTERED VIA KEYBOARD
KRA	SCALING CONSTANT ENTERED VIA KEYBOARD
KUF2	SCALING CONSTANT ENTERED VIA KEYBOARD
S0	SENSE LINE ZERO
RRH	REAL VALUE OF RH (NOT SCALED)

THE FOLLOWING ARE HYBRID SYSTEM SUBROUTINES

QMON	MONITOR MAGNETIC TAPE HANDLER
QRBADS	INPUTS A BLOCK OF A/D VALUES, SCALED
QRSLL	READS SENSE LINES
QSHYIN	IC HYBRID SYSTEM
QSIC	PLACES ANALOG COMPUTER IC
QSOP	PLACES ANALOG COMPUTER IN OPERATE
QSRUN	PLACES ANALOG LOGIC IN RUN MODE
QSSECN	PLACES ANALOG COMPUTER IN NORMAL SECOND (REAL TIME)
QSTDA	OUTPUTS D/A VALUE
QWBDAS	OUTPUTS A BLOCK OF D/A VALUES, SCALED
QWJDAS	TRANSFERS ONE D/A, SCALED

APPENDIX C

FORTRAN PROGRAM LISTING

THIS APPENDIX CONTAINS THE PACER 100 HYBRID PROGRAM LISTING.
THIS HYBRID PROGRAM WAS CODED IN FORTRAN IV, FLOATING POINT.
THE VARIABLES FOR THE PROGRAM ARE DEFINED IN APPENDIX B. THE FLOW
CHARTS FOR THE PROGRAM ARE ILLUSTRATED IN APPENDIX A.

\$JOB
\$FOR

MAIN PROGRAM, FPDPDP

DFDC FLOATING POINT HYBRID PROGRAM

MODE = 1, 2, OR 3 FOR HDG, ILS, OR TACAN, RESP.

SENSE SWITCHES 1 = ABORTS RUN
2 = TO CHANGE MODE
3 = RUN FREE, TIME INTERVAL NOT FIXED
4 = FORCE EARLY CAPTURE -- LAT, AXIS
5 = FORCE EARLY CAPTURE -- G/S
6 = D/A DATA CONTROL
7 =
8 = SIMULATION CONTROL, ACCEPT DATA

D/A,S INVERT; A/D,S DO NOT.

	ILS	TACAN
	---	----
DAC 0	= RH ERROR, 10 NM	= PH ERROR, 10 NM
1	= CTW, 100 KNOTS	= CTW, 100 KNOTS
2	= ATSW, 100 KNOTS	= LATSW
3	= GH ERROR	= GH ERROR
4	=	= -QDB
5	= VPTR	= VPTR
6	= GH ERROR	=
7	= HPTR	=

INTEGER GSSW, LATSW, IT, GSIT, MODE, MODELA, ITAPE, IDA
LOGICAL SENSW, SO, FTIME, NABLS, GSCORT
DIMENSION ADV(18), DAV(8), ADNA(2), D(54)
DIMENSION TRANSL(4,4), PL(4,4), XKL(4), KL(4), QL(4),
S RG(3), TRANSG(4,4), PG(4,4), XKG(4), KG(4), QG(4)
DIMENSION XR(4), XM(4)
REAL LIMIT
REAL ICOUNT
REAL JO, L, JO4, KHH, KG, KGD
REAL KL, KQ2, KQ3, KQ4, KRL
REAL KA, KB, KAY
REAL LA, LC, KY
REAL KHC, LLATSW
COMMON X, PHI, HO, Z, TH, GGAM, GM, TDM, XD,
SRL, QNF, T, QL, TRANSL, KQ2, KQ3, KQ4,
1ZK, YD, Y, QM, RH, QDB, RA,
1QH, CTE, CTW, ATE, ATW, WH, OH, VH, RN, PHIC,


```

SPSI,GAM,GAME,GAMH,GHE,GDE,PER,
SRANGE,ICOUNT,FK,KL,XKL,PL,DUMMY,
$ PHICT,PHICL,QLD2,IQH,ICTW,RHO,LC,G,
$ PHII,PHOLD,JO,IDA,IT
DATA ADNA(1),ADNA(2)/4HDFDC,4HDA /
DATA MODE / 2 /
DATA ITAPE,XJO,LLATSW,KHD,IICTW,IIRH,IQIH,IQDB,IIDA
$ / 0, 0.11, 1.0, 1.0, 1, 0, 0, 0, 0 /
DATA ISH,IQH,IGDB,IATSW,N5
$ / 0. 0, 0, 1, 9999 /
DATA KRL / 0.0005 /
DATA RG(1),RG(2),RG(3) / 0.00025, 0.0005, 0.0001 / ,

```

C PRESETS

```

ICTW = IICTW
IRH = IIRH
IQH = IIQH
IQDB = IIQDB
IDA = IIDA
GSA = 0.04363323
JO = XJO
G = 68625.36798
PI = 3.141592653
DUMMY = 0.0
N=4

```

```

CALL QSHYIN(IERR,680)

```

```

CALL QMON(22,'16)
CALL QMON(14)
CALL QMON(15)
CALL QMON(15)
CALL QMON(14)

```

```

TYPE 908
ACCEPT 909, ADNA(1), ADNA(2)

```

```

1 CONTINUE
CALL QSCLR(IERR)
CALL OSRUN(IERR)
CALL QSSTOP(IERR)
CALL QSSECN(IERR)

```

```

CALL QSIC(IERR)

```

```

CALL QMON(23,ADNA(1),3,'220','16)

```

```

IF ( SENSW(8) ) GO TO 7
TYPE 932
ACCEPT 901, ICNTRL
IF ( ICNTRL .LT. 1 .OR. ICNTRL .GT. 5 ) GO TO 1
GO TO ( 2, 3, 4, 5, 6 ), ICNTRL

```

```

2 CONTINUE
TYPE 913
ACCEPT 901, ICTW

```

TYPE 914
ACCEPT 901, IRH
TYPE 915
ACCEPT 901, IGH
TYPE 916
ACCEPT 901, IGDB
GO TO 7

3 CONTINUE
TYPE 930
ACCEPT 923, KRL
GO TO 7

4 CONTINUE
TYPE 917
ACCEPT 901, ISH
TYPE 918
ACCEPT 901, IGH
TYPE 919
ACCEPT 901, IGDB
TYPE 920
ACCEPT 901, IATSW
GO TO 7

5 CONTINUE

GO TO 7

6 CONTINUE
TYPE 905
ACCEPT 923, JO
TYPE 910
ACCEPT 901, ITAPE
TYPE 931
ACCEPT 901, IDA
TYPE 907
ACCEPT 901, NS

7 CONTINUE

T = JO/3600.0

C INITIALIZE TRANSITION MATRICES
TRANSL(3,3) = EXP(-900.0*T)
TRANSL(4,4) = EXP(-3600.0*T)
TRANSG(3,3) = EXP(-900.0*T)
TRANSG(4,4) = EXP(-1800.0*T)

C XXXXXX POWER CLEAR
ICOUNT=0.0
X = 5.0

C ... INITIALIZE STACKS S,
MODEL = 0
PHOLD = 0.0

DO 75 I = 1, 8
DAV(I) = 0.0

```

75  CONTINUE
    CALL QWRDAR ( DAV, 0, 8, IERR )
    CALL QSTDA

C   INITIALIZE CONSTANT ELEMENTS OF ALL MATRICES
DO 77 I = 1, 4
    XI = 0.0
    XR(I) = 0.0
    XM(I) = 0.0
    KL(I) = 0.0
    KG(I) = 0.0
    YKL(I) = 0.0
    YKG(I) = 0.0
    DO 77 JI = 1, 4
        PL(I,JI) = 0.0
        PG(I,JI) = 0.0
77  CONTINUE

    CALL QSIC(IERR)
    ATW=0.0
    TD=0.0
    I4=0

    CALL QRSLL( 0, S0, IERR )
8   CALL QRSLL( 0, S0, IERR )
    IF ( .NOT. S0 ) GO TO 8

    CALL QSOP( IERR )
    GO TO 10

C *** START OF PROGRAM LOOP
9   CONTINUE
    CALL QRSLL( 0, S0, IERR )
    IF ( SENSW(3) ) GO TO 10
    IF ( .NOT. S0 ) GO TO 9
10  CONTINUE

    IF ( SENSW(1) ) GO TO 500
    ABORT RUN
    IF ( ( MODE .EQ. 2 ) .AND. ( X .LT. 0.16667 ) ) GO TO 500
    IF ( I4.GT.N5 ) GO TO 500

    IF ( .NOT. SENSW(2) ) GO TO 11
    TYPE 900
    ACCEPT 901, MODE
11  CONTINUE
    FTIME = .FALSE.
    IF ( MODELA .EQ. MODE ) GO TO 12
    FTIME = .TRUE.
    MODELA = MODE
    GOTO RSTART S,
    LATSW = 1
    NABLGS = .FALSE.
    GSCORT = .FALSE.
    GSSW = 4
    HDGOR = 0.0
C
12  CONTINUE

```



```

C INPUT CONVERSION
  CALL QRBADR( ADV, 0, 18, IERR )
  IF ( MODE .NE. 2 ) GO TO 13
C ILS ANALOG VALUES ( RADIANS, KNOTS, NM )
  X = 20.*ADV(1)
  PSI = 3.4906585*ADV(5)
  Y = ADV(2)
  QM = 0.06981317*ADV(3)
  Z = ADV(8)/3.035
  TM = 1.7453293*ADV(9)
  GGAM = 0.174532925*ADV(10)
  GM = 0.024467233*ADV(11)
  TDM = 1.7453293*ADV(12)
  YD = -1000.0*ADV(13)
  XD = -1000.0*ADV(14)
  QNF = -0.717+0.06981317*ADV(15)
  GNF = -0.1357+0.024467233*ADV(16)
  RANGE = SQRT( (X+1.5)**2 + Y**2 )
  GO TO 14

13 CONTINUE
C TACAN ANALOG VALUES ( RADIANS, KNOTS, NM )
  X = 200.0*ADV(1)
  Y = 20.0*ADV(2)
  QM = 0.465421134*ADV(3)
  PSI = 3.4906585*ADV(5) = HDGOB
  GNF = -0.465421134*ADV(15)
  RHQ = RH
  RH = SQRT( X**2 + Y**2 )
  RANGE = RH
C MUST QUANTIZE DME RANGE

14 CONTINUE

  HH = 3.4906585*ADV(17)
  GAM = -3.4906585*ADV(4)
  PHI = 1.74532925*ADV(6)
  UO = 1000.0*ADV(7)

  GO TO ( 100, 200, 400 ), MODE
C HDG ILS TACAN
C GOTO MODE SWITCH(MODE)

C MAN.HDG.COMP( PSIMH )
100 CONTINUE
  AED1 = HH - PSI
  PHIC = XMANHD( AED1, UO )
  IT = 0
  CALL LATFIN
  VPTR = -0.572957795*PHIC
  IF ( IDA .EQ. 1 ) VPTR = -0.572957795*( PHIC - PHI )
  CALL QWJDAR( VPTR, 5, IERR )
  GO TO 9
C GOTO WAITLOOP

```

```

C   ILS MODE
200 CONTINUE
    IF ( .NOT. FTIME ) GO TO 201
        RH = 5.0
        RA = RH
        AY = 0.1
        CALL LATIC
        LC = .01
        PL(1,1) = 0.000006
        PL(2,2) = 0.000006
        PL(3,3) = 0.0000004
        PL(4,4) = 0.4
        QLO2 = 1.0 - EXP( -3600.0*T )
        QL(3) = 0.000000136*( 1.0 - EXP(-1800.0*T) )
        QL(4) = 5.8232845E03*( 1.0 - EXP(-4311.377*T) )
C   THE FOLLOWING IS FOR GS ALT. HOLD MODE
        AH = Z
        TF1P = TM
        HF15 = Z

C   ILS.COMMON
201 CONTINUE
    IF ( IT .GT. 0 ) GO TO 202
        XKL(1) = QM
        XKL(2) = -UO*SIN( PSI - QM )/RH
        QDB = XKL(2)

202 CONTINUE
    IF ( ABS(QH) .LT. 0.062831853 ) RA = VH*G*SIN( QH - GAMH )/QDB
    AY=0.0
    IF ( T4.LT.400.0 ) AY=0.1*EXP(-T4/8.0)
    RHO=RH
    RE = RH - T*VH*G*COS(GAMH)
    RD = RA - RE
    RH = RE + AY*LIMIT( 20.0*JO, RD )
    IF ( RH .LT. 5.0 ) RH = 5.0
    IF ( GSCORT ) RH = RE + 0.1*( SH + 1.5 - RE )

    IF ( IRH .EQ. 1 ) RH = RANGE

        RH1 = RH
        RH2 = 1.0
        SQM = QM
    CALL LATCOM( RH1, RH2, SQM )

    IF ( IQDB .EQ. 1 ) QDB = -UO*SIN( GAM - QNF )/RANGE

    KY=8.0/(8.0+572.96*ABS(QH))
    PHICT = ( 347.4*KY*QH + 0.8333*QDB )*.0174532925*1.25*5.0

    QME = QM - QNF
    QHE = QH - QNF
    QDE = QDB - YD/RANGE
    RER = RH - RANGE

    GO TO ( 210, 215, 220 ), LATSW
        HH CAPT TRACK

```

```

C   HEADING HOLD
210  CONTINUE
    IC = 0.01
    AED1 = HH = PSI
    PHIC = XMANHD( AED1, UO )
C   LOC.BEAM.SENSOR.ETC
    IF ( SENSW(4) ) GO TO 213
    RL = 0.0
    IF ( ABS(QH) .LE. 0.055850536 ) GO TO 211
    ATW = 0.0
    CTW = 0.0
    IT = 0
    RH = 5.0
    GO TO 225
211  CONTINUE
    RL = 0.00000001
    IF ( IT .EQ. 30 ) GO TO 212
    IT = IT + 1
    ATW = 0.0
    CTW = 0.0
    GO TO 225
212  CONTINUE
    RL = 0.00000001
    IF ( ( ( RH.GT.10.0 ) .AND. ( ABS(QH).GT.0.0418879 ) ) .OR.
$      ( ( RH.LE.10.0 ) .AND. ( ABS(QH).GT.(0.0418879+0.001396263*
$      (10.0-RH))) ) ) GO TO 225
    IF ( 0.0069974 .GT. ABS(QH) ) GO TO 213
    IF ( ABS(PHICL) .LT. ( 2.0/57.3 + ABS(2.0*QH) ) ) GO TO 225
C   XFER TO CAPTURE
213  CONTINUE
    IT = 30
    RL = 0.000000001
    LATSW = 2
    NABLGS = .TRUE.

C   LCAPTURE.COMP
215  CONTINUE
    RL=0.00000001
    LC = 0.02
    PHIC = PHICL
C   LOC.CAP.TO.TRACK
    IF ( ( ABS(QH) .GT. 0.0069974 ) .AND.
$      ( (ABS(QH).GT.0.01047) .OR. (ABS(GAMH).GT.0.349066)) ) GO TO 225
216  CONTINUE
    RL = 0.0000001
    LATSW = 3
    HPHIC = PHIC = PHICT

C   LTRACK.COMP
220  CONTINUE
    T4=T4+1.0
    LC = 0.015
    AY = 0.8825*AY
    PHIC = HPHIC + PHICT
    HPHIC = 0.985*HPHIC
    IF ( RH .LE. 5.0 ) PHIC = RH*PHIC/5.0

```


C CTW = 0.0

```
XH(1) = XM(1) + QME
XM(2) = XM(2) + QME
XM(3) = XM(3) + QNF
XM(4) = XM(4) + (PHIC-PHI)
XR(1) = XR(1) + (QME)**2
XR(2) = XR(2) + (QME)**2
XR(3) = XR(3) + (QNF)**2
XR(4) = XR(4) + (PHIC-PHI)**2
XI = XI + 1.0
```

```
225 CONTINUE
CALL LATFIN
C GOTO WAITLOOP
```

```
C GS,INIT
IF ( ( (GM) .GT. 0.009773840 ) .OR. ( .NOT. NABLGS ) )
  GO TO 350
IF ( GSSW .NE. 4 ) GO TO 301
300 CONTINUE
```

```
GSSW = 1
GST = 0
GH = GM
SH = Z/(GSA - GH)
ATGSN = GSA + TM
GDH = -0.7*UO*SIN(ATGSN)/SH
GDF = GDH
VHTN = UO
UF1 = UO
ATSW = 0.0
UF2 = 0.0
TF15 = TM
TF1 = TM
```

```
C IC XK MATRICS
CALL KALCLR( XKG, PG )
XKG(1) = GM
XKG(2) = GDH
```

```
C IC P MATRICS
PG(1,1) = 0.0034084
PG(2,2) = 112.89
PG(3,3) = 0.0034084
PG(4,4) = 1164000.0
QG(2) = 1.0 - EXP( -3600.0*T )
QG(2) = 20.0*QG(2)
QG(3) = 0.00815575*( 1.0 - EXP( -1800.0*T ) )
QG(3) = 0.05*QG(3)
QG(3) = 7.5*QG(3)
QG(4) = 1.1646569E07*( 1.0 - EXP( -3600.0*T ) )
QG(4) = 20.0*QG(4)
```

```
C GS,COMMON
301 CONTINUE
A15 = EXP( -JO )
IF ( OSCORT ) A15 = EXP( -JO/15.0 )
```

```

TF15 = A15*( TF15 - TM ) + TM
TF15C = TM - TF15
AED1 = EXP( -0.015915494*JO )
UF1 = AED1*( UF1 - UO ) + UO
UF1C = UO - UF1
AED1 = EXP( -0.15915494*JO )
UF2 = AED1*( UF2 - UF1C ) + UF1C
SHO = SH
SA = Z/( GSA - GH )
SE = SH - T*VHTN*COS(ATGSN)
SH = 0.9*SE + 0.1*SA

SHI = SQRT( X**2 + Z**2 )

IF ( ISH .EQ. 1 ) SH = SHI

IF ( 0.0 .GT. SH ) SH = 0.0
AED1 = EXP( -JO )
TF1 = AED1*( TF1 - TM ) + TM
TF1C = TM - TF1
TDM = TF1C
ATGSE = ATGSN + JO*TDM
ATSE = SH - SHO + T*VHTN*COS(ATGSE)
ATSW = ATSW - 0.008*LLATSW*ATSE/T
IF ( IATSW .EQ. 1 ) ATSW = 0.0
WSH = ATSW
VHTN = UO + ATSW
FK = -3600.0*VHTN*TDM/SH
DRH = 1.0
ZK = GM
IF ( GSIT .NE. 0 )
S CALL KALFIL( T, ZK, TRANSG, OG, XKG, PG, KG, RG(GSSW), FK, DRH)
GSIT = 1
GH = XKG(1)
GDH = XKG(2)

IF ( IGH .EQ. 1 ) GH = GM
IF ( IGDH .EQ. 1 ) GDH = -( GSA + TM )*UO/SHI

AED1 = EXP( -0.5*JO )
GDF = AED1*( GDF - GDH ) + GDH
ATGSN = -GDF*SH/( UO + ATSW )
TDCA = 10.59435*SH*GH*( 1.0 - VHTN/360.0 )
TDCA = TDCA + 1.5*0.57295*1.68614*SH*GDH/VHTN
TDCA = TDCA + 0.5*GDF*SH/VHTN
TDCA = TDCA - TF15C

GO TO ( 310, 315, 320, 350 ), GSSW
C PRECAP CAP TRK IDLE

C GS.PRECAP.COMP
310 CONTINUE
C GS.P.TO.CAP
IF ( SENSW(5) ) GO TO 311
IF ( ( GH ) .GT. 0.002850705 ) GO TO 350
311 CONTINUE
GSSW = 2

```

GSCORT = .TRUE.
TC = TM

C GS.CAP.COMP

315 CONTINUE

RPN = SH*GH/(1.0 - COS(ATGSN))
RPN = SIGN(RPN, ATGSN)
TDC = -VHTN/(3600.0*RPN)
TC = TC + JO*TDC
HPTR = 4.0*(TC - TM) - 0.02*UF2

C GS.C.TO.TRK

IF ((GH).GT. 0.001628973) GO TO 325
GSS4 = 3
TDCAT = HPTR - TDCA

C GS.TRACK.COMP

320 CONTINUE

HPTR = TDCA + TDCAT
TDCAT = 0.985*TDCAT

325 CONTINUE

C OUTPUT DA VALUES

DAV(2) = -.256478897*(GAMH-GAM)
DAV(3) = -.256478897*OH
DAV(5) = -14.32394489*(QH-QNF)
IF (SENS*(7)) DAV(5) = -14.32394489*QH
DAV(7) = -0.5*QDE

C OUTPUT D/A VALUES

DAV(1) = 0.05*(RH - RANGE)
IF (SENS*(6)) DAV(1) = 0.05*(SH - SHI)
DAV(2) = -0.005*WH
IF (SENS*(6)) DAV(2) = -0.005*ATSW
DAV(3) = (GH - GM)/0.0681317
IF (SENS*(6)) DAV(3) = -5.729577955*ATGSN
DAV(4) = -14.323945*(GH - QNF)
IF (SENS*(6)) DAV(4) = -40.8709886*(GH - GM)
DAV(5) = QDB - YD/RANGE
IF (SENS*(6)) DAV(5) = GDH - UD*GGAM/SHI
DAV(6) = -0.572957795*PHIC
IF (IDA.EQ. 1) DAV(6) = -0.572957795*(PHIC - PHI)
DAV(8) = -0.0572957795*HPTR

DO 340 I = 1, 8

DAV(I) = LIMIT(0.99999, DAV(I))

340 CONTINUE

CALL GWBDAR(DAV,0,8,IERR)

ICOUNT=ICOUNT+1.0

IF (ITAPE.EQ. 1) CALL QMON(16,RANGE,DUMMY)
I4=I4+1

CALL QSTDA
GO TO 9


```

C      END OF ILS PROGRAM LOOP

C      GS IDLE STATE
350  CONTINUE
    IF ( SENS*(5) .AND. NABLGS .AND. GSSW .EQ. 4 ) GO TO 300
    AED1 = EXP( -JO )
    TF1P = AED1*( TF1P - TM ) + TM
    TF1PC = TM - TF1P
    AED1 = EXP( -JO/15.0 )
    HF15 = AED1*( HF15 - Z ) + Z
    HF15C = Z - HF15
    HPTR = 2.0*( AH - Z ) - TF1PC - 2.0*HF15C - KHD*0.2*GGAM
    GO TO 325

400  CONTINUE
C      TAC.INIT
    IF ( .NOT. FTIME ) GO TO 401
    CALL LATIC
    PSINQW = GAMH
    PL(1,1) = 48.73
    PL(2,2) = 1000000.0
    PL(3,3) = 0.0625
    PL(4,4) = 4700000000.0
    QLQ2 = 47000000.0*( 1.0 - EXP( -360.0*T ) )
    QL(3) = 3.0*0.0001354*( 1.0 - EXP( -1800.0*T ) )
    QL(4) = 47000000000.0*( 1.0 - EXP( -3600.0*T ) )
    LC = 0.001
    RL = 500.0

    TRKTRP = 15.0/(15.0*57.2957795)
    GY = 5.0/57.2957795
    GYD = 0.12/57.2957795
C      HYBRID ONLY

C      TAC.COMMON
401  CONTINUE
    IF ( IT .GT. 0 ) GO TO 402
    XKL(1) = QM*RH
    XKL(2) = -UO*SIN( PSI )
    X2L = XKL(2)
    XSIL = GAMH
402  CONTINUE
    RH1 = 1.0
    RH2 = RH
    SQM = QM
    CALL LATCOM( RH1, RH2, SQM )
    IF ( IQDB .EQ. 1 ) QDB = -UO*SIN( GAM )
    GP = 1.0
    IF ( RH .GT. 50.0 ) GP = 50.0/RH
    QVSTR = 10.0*UO/500.0
    CF = EXP( -JO/30.0 )
    X2L = CF*( X2L - QDB ) + QDB
    XSIL = CF*( XSIL - GAMH ) + GAMH
    YDT = X2L - VH*G*( GAMH - XSIL )
    AED1 = XKL(1)
    IF ( IQH .EQ. 1 ) AED1 = QM*RH

```

PHICT = GP*(GV*AED1 + GYD*YDT)

GO TO (410, 415, 420, 425), LATSW

C TACMH.COMP

410 CONTINUE

AED1 = HH - PSI

PHIC = XMANHD(AED1, UO)

C TACMH.TO.CAP

IF (SENSX(4)) GO TO 411

IF ((RH*ABS(QH) .GT. 10.0) .AND. (IT .LT. 1)) GO TO 430

IT = IT + 1

IF (IT .LT. 30) GO TO 430

ADJPHL = 0.35 + LIMIT(0.15, 0.3*(RH-70.0)/80.0)

IF (((PHICL*PHICT .LE. 0.0) .OR. (ABS(PHICT) .LE.

\$ ABS(PHICL*ADJPHL))) .OR.

\$ ((ABS(QH) .LE. 0.01746) .AND.

\$ (ABS(PHICT) .GT. 0.01746))) GO TO 430

411 CONTINUE

IT = 30

LATSW = 2

C TACAPTURE.COMP

415 CONTINUE

RL = 500.0

PSINOW = PSI

IF ((ABS(PHICL) .LE. ABS(PHICT)) .AND.

\$ (QH*PHICT .LT. 0.0)) GO TO 416

PHIC = PHICT

C ITEST = .TRUE.

C TACAP.TO.TRK

IF (ABS(QH) .GT. TRKTRP) GO TO 430

LATSW = 3

GO TO 420

416 CONTINUE

PHIC = PHICL

C ITEST = .FALSE.

GO TO 430

C TAC.TRACK.COMP

420 CONTINUE

PHIC = PHICT

RL = LIMIT(2500.0, 25.0*RH)

RL = KRL*RL

PSINQW = 0.90484*(PSINOW - PSI) + PSI

QME = QM - QNF

QHE = QH - QNF

QDE = QDB - YD/RANGE

RER = RH - RANGE

XM(1) = XM(1) + QME

XM(2) = XM(2) + QHE

XM(3) = XM(3) + QNF

XM(4) = XM(4) + (PHIC-PHI)

XR(1) = XR(1) + (QME)**2

XR(2) = XR(2) + (QHE)**2

XR(3) = XR(3) + (QNF)**2

XR(4) = XR(4) + (PHIC-PHI)**2
XI = XI + 1.0

C TTRK.TO.OVS
IF (RH .GT. OVSTP) GO TO 430
LATSW = 4

C OVS.TO.CAP
425 CONTINUE
AED1 = HH - PSI
PHIC = XMANHD(AED1, UO)
IF (RH .LE. OVSTP) GO TO 430

C *** MUST RE-IC
LATSW = 2
PL(1,1) = 48.73
PL(2,2) = 1000000.0
PL(3,3) = 0.0625
PL(4,4) = 470000000.0
XKL(1) = QM*RH
XKL(2) = -UO*SIN(PSI + PSINOW)
X2L = XKL(2)
XKL(3) = 0.0
XKL(4) = 0.0
CTW = 0.0
GAMH = PSI
XSIL = PSI

430 CONTINUE
CALL LATFIN

DAV(1) = -QM
DAV(2) = -0.005*CTW
XXXX = LATSW
DAV(3) = 0.1*XXXX
DAV(4) = (QH - QM)
DAV(5) = -QDR*0.572958
DAV(6) = -0.572957795*PHIC
IF (IDA .EQ. 1) DAV(6) = -0.572957795*(PHIC - PHI)

DO 431 I = 1, 8
DAV(I) = LIMIT(0.99999, DAV(I))

431 CONTINUE

CALL QWBDR(DAV, 0, 8, IERR)

IF (ITAPE .EQ. 1) CALL QMON(16, X, DUMMY)

CALL QSTDA
GO TO 9

C END OF TACAN PROGRAM LOOP

500 CONTINUE

CALL QSIC(IERR)

CALL QSDLY(15)
CALL QMON(24, 116)


```

CALL VPRNT (XKL,N,N,6HXKL )
CALL PRNT (PL,N,N,6HPL )
CALL VPRNT (KL,N,N,6HKL )
AED1 = RH - RANGE
TYPE 933, AED1
TYPE 934, CTW

IF ( MODE .EQ. 3 .OR. GSSW .EQ. 4 ) GO TO 501
CALL VPRNT( XKG, N, N, 6H XKG )
CALL PRNT( PG, N, N, 6H PG )
CALL VPRNT( KG, N, N, 6H KG )
TYPE 935, ATSW

```

501 CONTINUE

```

IF ( SENSW(1) ) GO TO 1
IF ( LATSW .LT. 3 ) GO TO 9993
DO 9991 I=1,4
XM(I)=XM(I)/XI
9991 XR(I)=SQRT((XR(I)-(XM(I))*2)/XI)
AED1 = 4297.183462
IF ( MODE .EQ. 3 ) AED1 = 859.333925
DO 9992 I=1,3
XM(I) = AED1*XM(I)
XR(I) = AED1*XR(I)
9992 CONTINUE
XM(4)=57.2957795*XM(4)
XR(4)=57.2957795*XR(4)
CALL VPRNT(XR,N,N,6HX PMS )
CALL VPRNT(XM,N,N,6HX MEAN)
TYPE 940

```

9993 CONTINUE

```

CALL QMON(24,'16)
IF ( ITAPE .EQ. 0 ) GO TO 1
995 CALL QMON(21,ADNA(1),1,'16)
1000 TYPE 903
ACCEPT 904, IT1, IT2, IT3, IT4, IT5, IT6
TYPE 950
ACCEPT 901, N1
IF (N1.GT.1) GO TO 2000
IF(N1.LT.1) GO TO 1000
IF (IT1.LT.1) GO TO 1
CALL QMON(21,ADNA(1),1,'16)
DO 1001 J9=1,N4
CALL QMON(17,D(1),D(54))
TYPE 941, D(IT1),D(IT2),D(IT3),D(IT4),D(IT5),D(IT6)
IF (SENSW(1)) GO TO 1000
1001 CONTINUE
GO TO 1000
1000 CALL QMON(21,ADNA(1),1,'16)
DO 2001 J9=1,N4
CALL QMON(17,RANGE,DUMMY)
TYPE 942, ICOUNT,FK,SDUM1,SDUM3
CALL VPRNT(KL,N,N,6HKL )
CALL VPRNT(XKL,N,N,6HXKL )

```

```

CALL PRNT(PL,N,N,6HPL )
IF (SENSW(1)) GO TO 1000
2001 CONTINUE
GO TO 1000

900 FORMAT( 43H = 1, 2, OR 3 FOR HDG, ILS OR TACAN, RESP.)
901 FORMAT(I60)
902 FORMAT(/,2E15.8,/)
903 FORMAT(1H ,23H TYPE 6-2 DIGIT INTEGERS/)
904 FORMAT(6I2)
905 FORMAT( 10H = JO)
906 FORMAT( 8H = MODE)
907 FORMAT( 8H = NS)
908 FORMAT( 17H = FILE NAME)
909 FORMAT(2A4)
910 FORMAT( 47H = 0, 1, OR 2 - NO TAPE, SF, OR FT DAT, RESPT.)
911 FORMAT(10H ICOUNT = , 15, / )
912 FORMAT( 7H = INF)
913 FORMAT( 8H = ICTW)
914 FORMAT( 7H = IRH)
915 FORMAT( 7H = IGH)
916 FORMAT( 8H = IQDS)
917 FORMAT( 7H = ISH)
918 FORMAT( 7H = IGH)
919 FORMAT( 8H = IGDH)
920 FORMAT( 9H = IATSW)
921 FORMAT( 9H = IGGAM)
922 FORMAT( 13H = KATSW)
923 FORMAT( F20.10)
924 FORMAT( 13H = KTDCT)
925 FORMAT( 13H = KTDCC)
926 FORMAT( 12H = KU1F2)
927 FORMAT( 13H = KKCTW)
928 FORMAT( 11H = KFK)
929 FORMAT( 11H = KRA)
930 FORMAT( 11H = KRL)
931 FORMAT( 7H = IDA)
932 FORMAT( 49H = 1, 2, 3, 4, OR 5 FOR L, LK, G, GK, OR GENERAL)
933 FORMAT( 11H4 ERROR = , F14.8 )
934 FORMAT( 6HCTW = , F14.8 )
935 FORMAT( 7HATSW = , F14.8 )

940 FORMAT(1H ,1H /)
941 FORMAT(6F11.6)
942 FORMAT(4E12.6)

```

```

650 FORMAT(1H ,16H TYPE 1 TO ACCEPT/)
END
C SUB 1

```

```

SUBROUTINE LATCOM( RH1, RH2, SQM )
LOGICAL SENSW
REAL LIMIT
REAL JO,L,JO4,KHH,KG,KGD
REAL LC
REAL KL, KQ2,KQ3,KQ4
REAL ICOUNT

```

```

    DIMENSION TRANSL(4,4), PL(4,4), XKL(4), KL(4), QL(4),
$    RG(3), TRANSG(4,4), PG(4,4), XKG(4), KG(4), QG(4)
    COMMON X, PHI, UO, Z, TM, GGAM, GM, TDM, XD,
$    SRL, QNF, T, QL, TRANSL, KQ2, KQ3, KQ4,
$    IZK, YD, Y, RM, RH, QDR, RA,
$    IQH, CTE, CTW, ATE, ATW, WH, OH, VH, RN, PHIC,
$    SPSI, GAM, GAME, GAMH, QHE, QDE, RER,
$    SRANGE, ICOUNT, FK, KL, XKL, PL, DUMMY,
$    PHICT, PHICL, QLO2, IQH, ICTW, RHO, LC, G,
$    PHII, PHOLD, JO, ID4, IT
    QL(2) = QLO2*2.0
    IF ( IT .GT. 25 ) QL(2) = QLO2*0.0005

    DRH = RH2
    ZK = SQM
    FK = -G*SIN(PHI)*COS(GAMH)/(COS(PHI)*RH1)
    IF ( IT .GT. 0 )
$    CALL KALFIL( T, ZK, TRANSL, QL, XKL, PL, KL, RL, FK, DRH )
    QHO = QH
    QH = XKL(1)/RH2
    QDB=XKL(2)

    IF ( IQH .EQ. 1 ) QH = QM

    GAME = GAMH + T*G*SIN(PHI)/(VHG*COS(PHI))
    ATE=RH-RHO+ T*VHG*COS(GAME)
    CTE = RH*( QHO - QH ) - T*VHG*SIN(GAME)
    CTE = LIMIT( 0.0012, CTE )
    CTW = CTW + LC*CTE/T
    IF ( ICTW .EQ. 1 ) CTW = 0.0
    ATW=0.0
    WH=SQRT(ATW**2+CTW**2)
    OH=ATAN(CTW/ATA)
    SIGA=-1.0
    IF (CTW.GT.0.0) SIGA=+1.0
    IF (ATW.LT.0.0) OH=SIGA*(3.141592654-ABS(OH))
    CWH = WH*COS( OH - PSI )
    SWH = WH*SIN( OH - PSI )
    VH = SQRT( SWH**2 + ( UO + CWH )**2 )
    GAMH = PSI + ATAN( SWH/( UO + CWH ) )
    RN = RH*ABS(QH)
    AED1 = VH*VHG*( 1.0 - COS(GAMH) )
    AED1 = -SIGN( AED1, GAMH )
    PHICL = ATAN( AED1/( RN*G ) )
    IF (SENSW(4)) PHICL=-ATAN(RH*QDB**2/(G*QH*(1.0+COS(GAMH))))
    RETURN
    END
    SUB2

```

```

    SUBROUTINE KALFIL( T, ZK, TRANS, Q, XK, P, K, R, FK, H )

```

```

    ***** KALMAN FILTER SUBROUTINE *****

```

```

    REAL K
    DIMENSION TRANS(4,4), PP(4,4), K(4), DUM(4,4), Q(4),
    1    XK(4), XKP(4), P(4,4), CDUM(4)

```



```

C***** COMPUTE PP=TRANS*P*TRANST+Q *****
C
C
C   COMPUTE (DUM) =(TRANS)(P)
C
C   DUM(1,1)=P(1,1)+T*P(2,1)
C   DUM(1,2)=P(1,2)+T*P(2,2)
C   DUM(1,3)=P(1,3)+T*P(2,3)
C   DUM(1,4)=P(1,4)+T*P(2,4)
C   DUM(2,1)=P(2,1)+T*P(4,1)
C   DUM(2,2)=P(2,2)+T*P(4,2)
C   DUM(2,3)=P(2,3)+T*P(4,3)
C   DUM(2,4)=P(2,4)+T*P(4,4)
C   DUM(3,1)=TRANS(3,3)*P(3,1)
C   DUM(3,2)=TRANS(3,3)*P(3,2)
C   DUM(3,3)=TRANS(3,3)*P(3,3)
C   DUM(3,4)=TRANS(3,3)*P(3,4)
C   DUM(4,1)=TRANS(4,4)*P(4,1)
C   DUM(4,2)=TRANS(4,4)*P(4,2)
C   DUM(4,3)=TRANS(4,4)*P(4,3)
C   DUM(4,4)=TRANS(4,4)*P(4,4)
C
C   COMPUTE (PP)=(DUM)(TRANS)T+Q
C
C   PP(1,1)=DUM(1,1)+T*DUM(1,2)
C   PP(2,1)=DUM(2,1)+T*DUM(2,2)
C   PP(3,1)=DUM(3,1)+T*DUM(3,2)
C   PP(4,1)=DUM(4,1)+T*DUM(4,2)
C   PP(1,2)=DUM(1,2)+T*DUM(1,4)
C   PP(2,2)=DUM(2,2)+T*DUM(2,4)+Q(2)
C   PP(3,2)=DUM(3,2)+T*DUM(3,4)
C   PP(4,2)=DUM(4,2)+T*DUM(4,4)
C   PP(1,3)=DUM(1,3)+TRANS(3,3)
C   PP(2,3)=DUM(2,3)+TRANS(3,3)
C   PP(3,3)=DUM(3,3)+TRANS(3,3)+Q(3)
C   PP(4,3)=DUM(4,3)+TRANS(3,3)
C   PP(1,4)=DUM(1,4)+TRANS(4,4)
C   PP(2,4)=DUM(2,4)+TRANS(4,4)
C   PP(3,4)=DUM(3,4)+TRANS(4,4)
C   PP(4,4)=DUM(4,4)+TRANS(4,4)+Q(4)
C
C***** COMPUTE K=PP*HT*(1/H*PP*HT) *****
C
C   COMPUTE ( DUM ) =(PP)*(H)T
C
C   DO 30 I=1,4
30   CDUM(I) = PP(I,1) + H*PP(I,3)
C
C   COMPUTE (H) ( (PP)(H)T )
C
C   SDUM1 = CDUM(1) + H*CDUM(3) + R
C
C   K(1)=CDUM(1)/SDUM1
C   K(2)=CDUM(2)/SDUM1
C   K(3)=CDUM(3)/SDUM1
C   K(4)=CDUM(4)/SDUM1
C

```

```

C
C***** COMPUTE P=PP-K*H*PP *****
C
      DO 40 I=1,4
      DO 40 J=1,4
40    P(I,J) = PP(I,J) - K(I)*( PP(1,J) + H*PP(3,J) )
C
C***** COMPUTE KALMAN ESTIMATES *****
C
C      XP(K)=TRANS(K,K-1)*X(K-1) + F(K-1)
C
C      XKP(1)=XK(1)+T*XK(2)
C      XKP(2)=XK(2)+T*(XK(4)+FK)
C      XKP(3)=TRANS(3,3)*XK(3)
C      XKP(4)=TRANS(4,4)*XK(4)
C
C      X(K)=XP(K) + K(K)*( Z(K)-H(K)*XP(K) )
C
C      Z(K)-H(K)*XP(K)
C
C      SDUM3 = H*( ZK - XKP(3) ) - XKP(1)
C
C      XP(K)+K(K)*(Z(K)-H(K)*XP(K))
C
C      XK(1)=XKP(1)+K(1)*SDUM3
C      XK(2)=XKP(2)+K(2)*SDUM3
C      XK(3)=XKP(3)+K(3)*SDUM3
C      XK(4)=XKP(4)+K(4)*SDUM3
C      RETURN
C      END
C      SUB3

SUBROUTINE LATFIN
  REAL LC
  REAL KL, KQ2,KQ3,KQ4
  REAL ICOUNT
  REAL JO, LIMIT, LC, PI, AED1
  DIMENSION TRANS(4,4), PL(4,4), XKL(4), KL(4), QL(4),
&    RG(3), TRANSG(4,4), PG(4,4), XKG(4), KG(4), QG(4)
  COMMON X,PHI,JO,Z,TM,GGAM,GM,TDM,XD,
&    SRL,GNF,T,QL,TRANS,KQ2,KQ3,KQ4,
&    IZK,YD,Y,GM,RH,QD9,RA,
&    IGH,CTE,CTW,ATE,ATW,WH,OH,VHG,RN,PHIC,
&    PSI,GAM,GAME,GAMH,DHE,QDE,RER,
&    SPAN,ICOUNT,FK,KL,XKL,PL,DUMMY,
&    PHIC1,PHICL,GLO2,IGH,ICTW,RHO,LC,G,
&    PHII,PHOLD,JO,IDA,IT
  PHII = PHIC - PHOLD
  AED1 = 0.08726*JO
  PHIC = PHOLD + LIMIT( AED1, PHII )
  PHIC = LIMIT( 0.53, PHIC )
  PHOLD = PHIC
  RETURN
  END
  SUB3

```

```

SUBROUTINE LATIC
REAL LC
REAL ICOUNT
REAL JO,L,JO4,KHH,KG,KGD
REAL KL,KQ2,KQ3,KQ4
DIMENSION TRANSL(4,4), PL(4,4), XKL(4), KL(4), QL(4),
S   RG(3), TRANSG(4,4), PG(4,4), XKG(4), KG(4), QG(4)
COMMON X,PHI,UO,Z,TH,GGAM,GM,TDM,XD,
$   SRL,QNF,T,QL,TRANSL,KQ2,KQ3,KQ4,
$   IZK,YO,Y,QM,RH,QDR,RA,
$   IQH,CTE,CTW,ATE,ATW,WH,QH,VHG,RN,PHIC,
$   PSI,GAM,GAME,GAMH,QHE,QDE,RER,
$   SRANGE,ICOUNT,FK,KL,XKL,PL,DUMMY,
$   PHICT,PHICL,QLQ2,IQH,ICTW,RHO,LC,G,
$   PHII,PHOLD,JO,IDA,IT
CALL KALCLR( XKL, PL )
RL = 0.0
IT = 0
QH = QM
CTW = 0.0
VHG = UO
GAMH = PSI
PHOLD = PHI
C XXX PHILAG = 0.0
RETURN
END
C SUB5

SUBROUTINE KALCLR( XK, P )
DIMENSION XK(4), P(4,4)
DO 1 I = 1,4
XK(I) = 0.0
DO 1 J = 1,4
1   P(I,J) = 0.0
RETURN
END
C SUB6

FUNCTION LIMIT( L, VAL )
REAL L, LIMIT
LIMIT = VAL
IF ( ABS(VAL) .GT. L ) LIMIT = SIGN( L, VAL )
RETURN
END
C SUB7

FUNCTION XMANHO( PSI, UO )
REAL LIMIT
AED1 = ( UO - 350.0 )/150.0
GHH = 2.0 + LIMIT( 1.0, AED1 )
AED1 = LIMIT( 0.5235987, PSI )
XMANHO = GHH*AED1
RETURN
END
C SUB8

SUBROUTINE PRNT(A,NRA,NCA,ITITLE)

```



```

        DIMENSION A(4,4),ITITLE(3)
        TYPE 101,ITITLE(1),ITITLE(2),ITITLE(3)
        DO 95 I=1,4
95      TYPE 100,A(I,1),A(I,2),A(I,3),A(I,4)
        100 FORMAT(1X,(4E14.5))
        101 FORMAT(1X,/,5X,3A2,)
        RETURN
        END
C      SUB9

```

```

        SUBROUTINE VPRNT(A,NRA,NCA,ITITLE)
        DIMENSION A(4),ITITLE(3)
        TYPE 101,ITITLE(1),ITITLE(2),ITITLE(3)
        TYPE 100,A(1),A(2),A(3),A(4)
        100 FORMAT(1X,(4E14.5))
        101 FORMAT(1X,/,5X,3A2,)
        RETURN
        END

```

```

*
$FILE,14,QMON
$FILE,14,LINKN
$OUT,15,TKDFDL
$LOAD
$MON

```

APPENDIX D

FORTRAN PROGRAM LISTING (SCALED)

THIS APPENDIX CONTAINS THE PACER 100 HYBRID PROGRAM LISTING.
THIS HYBRID PROGRAM WAS CODED IN FORTRAN IV, SCALED FRACTION.
THE VARIABLES FOR THE PROGRAM ARE DEFINED IN APPENDIX B. THE FLOW
CHARTS FOR THE PROGRAM ARE ILLUSTRATED IN APPENDIX A.

```
#W,16
#C,16
#N,ORJTAA,16,1
#L,FORTR,14
#R,70000,20004,40706
#G,1000
C      MAIN PROGRAM, SFDFOP
C
C *****
C      DFDC SCALED FRACTION HYBRID PROGRAM
C *****
C
C      MODE = 1, 2, OR 3 FOR HDG, ILS, OR TACAN, RESP.
C
C      VPTR = -PHIC
C
C      SENSE SWITCHS
C          1 = ABORTS RUN
C          2 = TO CHANGE MODE
C          3 = RUN FREE, TIME INTERVAL NOT FIXED
C          4 = FORCE EARLY CAPTURE -- LAT. AXIS
C          5 = FORCE EARLY CAPTURE -- G/S
C          6 =
C          7 =
C          8 = SIMULATION CONTROL, ACCEPT DATA
C
C          ILS                      TACAN
C          ---                      -----
C          DAC 0 = RH ERROR, 10 NM    = RH ERROR, 10 NM
C              1 = CTW, 100 KNOTS      = CTW, 100 KNOTS
C              2 = ATSW, 100 KNOTS      = LATSW
C              3 = QH ERROR             = QH ERROR
C              4 =                     = -QDB
C              5 = VPTR                 = VPTR
C              6 = GH ERROR             =
C              7 = HPTR                 =
C
C      XXX NOTE KALFIL 1 AND 2 ARE NOT SCALED FRACTION
C
C      SCALED FRACTION LOK(160)
C      DIMENSION ADNA(2)
C      DIMENSION PL(16), QL(4), XKL(4), TRLAT(4), CDUM(4), PGS(16),
C      S      QGS(4), XKGS(4), TRGS(4), CDUMG(4), RG(4)
C      REAL SDUM, SDUMG, FK, ZK, RL
C      SCALED FRACTION SABS,SSQRT,SSIGN, SINX, COSX, ATANB,SLIMIT,XMANHD
C      SCALED FRACTION ADV(26), DAV(8)
C      SCALED FRACTION AH, HH, TMF1, HF15, TMF1C, HF15C, RHILS, QNF,
C      S      AEO3, GGAM, GAM
C      SCALED FRACTION TM, PHI, DMER, PSIEHH, PSIECR, QM, TQM, GM, UO, Z,
C      S      X, Y, SINPHI, COSPHI, RA, RE, RD, AY, RH, QH0, QH, QDB, GAME,
```


APPENDIX D

FORTRAN PROGRAM LISTING (SCALED)

```

$ CTE, KCTE, CTW, WH, OH, CWH, SWH, AED1, VMG, GAMH, PHICL, KY,
$ PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
$ KSCAL1, KSCAL2, LC, PHILAG, AED2,
$ YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
$ TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
$ HPTR, TDCA, TDCAV, TDCAT
LOGICAL SENSW, S0, FTIME, NABLS, GSCORT
INTEGER GSSW, LATSW, IT, GSIT, MODE, MODELA, ICOUNT, ITAPE
COMMON TM, PHI, DMR, PSIEHH, PSIECR, QM, TQM, GM, U0, Z,
$ RA, RE, RD, AY, RH, QH0, OH, QDB, GAME,
$ CTE, CTW, WH, OH, CWH, SWH, AED1, VMG, GAMH, PHICL, KY,
$ PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
$ RHILS,
$ QL, FK, RRH, QL02, TRLAT, XKL, SDUM, CDUM, PL, RL, IT,
$ DUMMY,
$ YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
$ TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
$ HPTR, TDCA, TDCAT, LC,
$ LATSW, FTIME,
$ XXXXX, ICTW
DATA ADNA(1), ADNA(2) / 4HDFDC, 4HDA /
DATA MODE / 2 /
DATA ITAPE, INF, IICTW, IRH, IATSW / 0, 1, 0, 0, 0 /
DATA TRGS(1), TRGS(2), TRGS(3) / 0.09752456, 0.0951125, 0.014222222 /
DATA RG(1), RG(2), RG(3) / 7.8175E-7, 1.5625E-6, 3.125E-6 /

```

CALL AFAULT

C PRESETS

```

ICTW = IICTW
QL(1) = 0.0
QGS(1) = 0.0
TRLAT(1) = 0.9753099
TRLAT(2) = 0.90483741

```

CALL QSHYIN(IERR, 680)

```

CALL QMON(22, '16)
CALL QMON(14)
CALL QMON(15)
CALL QMON(15)
CALL QMON(14)

```

CALL QSSSECN(IERR)

C SET TIME SCALE TO NORMAL SECONDS FOR THE ANALOG LOGIC

```

1 CALL QSRUN( IERR )
CALL QSSSTOP( IERR )

```

C TURN ANALOG LOGIC ON AND OFF

CALL QSIC (IERR)

C PLACE THE ANALOG COMPUTER IN IC MODE

CALL QMON(23, ADNA(1), 3, '230, '16)

IF (SENSW(8)) GO TO 7

```

TYPE 932
ACCEPT 901, ICNTRL
IF ( ICNTRL .LT. 1 .OR. ICNTRL .GT. 5 ) GO TO 1
GO TO ( 2, 3, 4, 5, 6 ), ICNTRL

2  CONTINUE
   TYPE 912
   ACCEPT 901, INF
   TYPE 913
   ACCEPT 901, IICTW
   ICTW = IICTW
   TYPE 914
   ACCEPT 901, IRH
   GO TO 7

3  CONTINUE
   GO TO 7

4  CONTINUE
   TYPE 920
   ACCEPT 901, IATSW
   GO TO 7

5  CONTINUE
   GO TO 7

6  CONTINUE
   TYPE 910
   ACCEPT 901, ITAPE

7  CONTINUE

C *** POWER CLEAR
   ICOUNT = 0
   X = 0.015
C ... INITIALIZE STACKS &,
   MODELA = 0
   PHOLD = 0.05
C   NEED FOR HDG.MODE
   DO 75 I = 1, 8
   DAV(I) = 0.05
75  CONTINUE
   CALL QWBDS( DAV, 0, 8, IERR )
   CALL GSTOA
C   ZERO DAC

   CALL QRSLL( 0, 50, IERR )
8   CALL QRSLL( 0, 50, IERR )
   IF ( .NOT. 50 ) GO TO 8

   CALL QSOP( IERR )
   GO TO 10

C *** START OF PROGRAM LOOP
9   CALL QRSLL( 0, 50, IERR )

```



```

      IF ( SENSW(3) ) GO TO 10
      IF ( .NOT. S0 ) GO TO 9
10    CONTINUE

      IF ( SENSW(1) ) GO TO 500
C     ABORT RUN
      IF ( ( MODE .EQ. 2 ) .AND. ( X .LT. 0.00528 ) ) GO TO 500

      IF ( .NOT. SENSW(2) ) GO TO 11
      TYPE 900
      ACCEPT 901, MODE
11    CONTINUE
      FTIME = .FALSE.
      IF ( MODELA .EQ. MODE ) GO TO 12
      FTIME = .TRUE.
      MODELA = MODE
C     GOTO RSTART S,
      LATSW = 1
      NABLGS = .FALSE.
      GSCORT = .FALSE.
      GSSW = 4
      HDGOB = 0.05
12    CONTINUE

C     INPUT CONVERSION
      CALL ORBADS( ADV, 0, 18, IERR )

      IF ( MODE .NE. 2 ) GO TO 13

C     SCALED ANALOG VALUES ** ILS ** ( RADIANS, KNOTS, NM )
C     -----
      TM * PI           = ADV(9) * 100.0/57.2957795
      PHI * PI          = ADV(6) * 100.0/57.2957795
      PSI * PI          = ADV(5) * 200.0/57.2957795
      QM * PI/32.0      = ADV(3) * 300.0/(57.2957795*75.0)
      GM * PI/128.0     = ADV(11) * 300.0/(57.2957795*214.29)
      UO * 840.0        = ADV(7) * 1000.0
      Z * 1.0           = ADV(8) * 2000.0 FT / 6076.1155 FT/NM
      X * 32.0          = ADV(1) * 20.0
      Y * PI            = ADV(2) * 1.0
C     DTR = 57.2957795
C     PI = 3.141592653

      TM = ADV(9)*0.555568
      QM = ADV(3)*0.711118
      QNF = -0.509878*ADV(15)
      GM = ADV(11)*0.995548
      GGAM = 0.055568*ADV(10)
      Z = ADV(8)*0.329768
      X = ADV(1)*0.6258
      Y = ADV(2)*0.318318
      RHILS = SSQRT( (X+0.046888)*(X+0.046888) + Y*Y*0.009648 )
      RRH = RHILS

      GO TO 14

```

```

13  CONTINUE
C
C      SCALED ANALOG VALUES ** TACAN ** ( RADIANS, KNOTS, NM )
C      -----
C      PHI * PI          = ADV(6) * 100.0/57.2957795
C      PSI * PI          = ADV(5) * 200.0/57.2957795
C      QM * PI/16.0      = ADV(3) * 400.0/(15.0*57.2957795)
C      U0 * 840.0        = ADV(7) * 1000.0
C      X * 512.0         = ADV(1) * 200.0
C      Y * 32.0*PI       = ADV(2) * 20.0
C
C      QM = ADV(3)/0.421875
C      QNF = -ADV(15)/0.588385
C      X = ADV(1)*0.390625
C      Y = ADV(2)*0.198945
C      RH = SSQRT( X*X + Y*Y*0.038555 )
C  MUST DISCRETIZE DME RANGE

14  CONTINUE
    HH = ADV(17)/0.95
    GAM = -ADV(4)/0.95
    PSI = ADV(5)/0.95
    PSIEMH = PSI
    PSIECR = PSI

    PHI = ADV(6)*0.555565
    U0 = ADV(7)/0.845

    GO TO ( 100, 200, 400 ), MODE
C      HDG ILS TACAN
C  GO TO MODE SWITCH(MODE)

C  MON.HDG.COMP( PSIEMH )
100  CONTINUE
    AED1 = HH - PSIEMH
    PHIC = XMANHD( AED1, U0 )
    IT = 0
    CALL LATFIN( 0.99999 )
    GO TO 9
C  GOTO WAITLOOP

C  ILS MODE
200  CONTINUE
    IF ( .NOT. FTIME ) GO TO 201
    RA = 0.156255
    RH = 0.156255
    AY = 0.15
    CALL LATIC
    LC = 0.371368
    PL(1) = 0.098364
    PL(6) = 0.000000375
    PL(11) = 0.0065536
    PL(16) = 0.0000000238418
    QL02 = 0.0119395325

```

QL(3) = 0.0001086717952
 QL(4) = 0.00003917558372
 TRLAT(3) = 0.01422222222

C XXXX THE FOLLOWING IS FOR GS ALT. HOLD MODE

AH = Z
 TMF1 = TM
 HF15 = Z

C ILS.COMMON

201 CONTINUE

IF (IT .GT. 0) GO TO 202

XKL(1) = QM

XKL(2) = -0.48735*U0*SINX(PSIECR - 0.031255*QM)/RH

QDB = XKL(2)

202 CONTINUE

IF (SABS(QM) .LE. 0.639995)

S RA = 0.48735*VHG*SINX(0.031255*QM - GAMH)/QDB

RE = RH - 0.000805*VHG*CO SX(GAMH)

RD = RA - RE

IF ((SABS(QM) .LT. 0.177295) .AND. (LATSW .NE. 1))

S AY = 0.955*AY

RH = RE + AY*SLIMIT(0.06255, RD)

IF (RH .LT. 0.156255) RH = 0.156255

IF (GSCORT) RH = RE + 0.15*(SH - RE + 0.046885)

IF (IRH .EQ. 1) RH = RHLS

AED1 = 0.006745

AED2 = 0.034585

CALL LATCOM(RH, 0.999995, QM, AED2, AED1, 0.170005, 0.999995)

KY = 0.102955/(0.110675 + 0.889335*SABS(QM))

PHICT = (KY*QM + QDB)/0.687335

GO TO (210, 215, 220), LATSW

C MHDGMS

210 CONTINUE

AED1 = RH - PSIECR

PHIC = XMANHD(AED1, U0)

C LOC.BEAM.SENSOR.ETC

IF (SENSW(4)) GO TO 212

RL = 0.0

IF (SABS(QM) .LE. 0.568895) GO TO 211

IT = 0

RH = 0.156255

GO TO 225

211 CONTINUE

RL = 0.00016384

IT = IT + 1

IF (IT .LT. 30) GO TO 225

RL = 0.00016384

IF (((RH .GT. 0.31255) .AND. (SABS(QM) .GT. 0.426675)) .OR.

S ((RH .LE. 0.31255) .AND. (SABS(QM) .GT. (0.568895 -

S 0.455095*RH))) GO TO 225

IF (0.071115 .GT. SABS(QM)) GO TO 212

IF (SABS(PHICL) .LE. (0.011115 + 0.06255*SABS(QM))) GO TO 225

212 CONTINUE

RL = 0.00016384

LATSW = 2

NABLGS = ,TRUE.

C LCAPTURE,COMP

215 CONTINUE

PHIC = PHICL

C LOC,CAP,TO,TRACK

IF (0.071119 ,GT. SABS(QH)) GO TO 216

IF ((0.106658 ,GT. SABS(QH)) ,AND. (0.111128

\$,GT. SABS(GAMH))) GO TO 216

IF ((-0.106658 ,GT. QH) ,AND. (GAMH ,GT. 0.111128))

\$ GO TO 216

IF ((GH ,GT. 0.106658) ,AND. (-0.111128 ,GT. GAMH))

\$ GO TO 216

GO TO 225

215 CONTINUE

LATSW = 3

HPHIC = PHIC - PHICT

RL = 0.0016384

C LTRACK,COMP

220 CONTINUE

AY = 0.88258*AY

PHIC = HPHIC + PHICT

HPHIC = HPHIC*0.998

IF (RH ,LE. 0.156258) PHIC = RH*PHIC/0.156258

C CTW = 0.08

225 CONTINUE

CALL LATFIN(0.99999)

C GOTO WAITLOOP

C THE FOLLOWING IS FOR PGM ERROR TRACKING

C MAG. OF RH ERROR = 10.0 NM

AED1 = (RH - RHIL8) /0.31258

CALL QWJDAS(AED1, 0, IERR)

C MAG. OF AED1 = (CTW =) 100.0 KNOTS

AED1 = CTW/0.119058

CALL QWJDAS(AED1, 1, IERR)

C MAG. OF AED1 = QH ERROR = 30UAMPS

AED1 = (QH - QM) /0.158

IF (INF ,EQ. 1) AED1 = (QH - QNF) /0.158

CALL QWJDAS(AED1, 3, IERR)

AED1 = -0.443008*UO*SINX(GAM - 0.031258*QNF) /RHIL8

AED1 = QDR - AED1

CALL QWJDAS(AED1, 4, IERR)

C GS,INIT

IF (((GM) ,GT. 0.398228) ,OR. (,NOT. NABLGS))

\$ GO TO 350

IF (GSSW ,NE. 4) GO TO 301

300 CONTINUE

GSSW = 1

GSIT = 0

GM = GM

SH = 0.458375*Z / (0.648 - 0.368*GM)

ATGSN = 0.013895 + TM

GDN = -0.365568*UO*SINX(ATGSN) /SH

```

GDF = GDH
VHTN = U0
UF1 = U0
ATSW = 0.0S
UF2 = 0.0S
TF15 = TM
TF1 = TM
CALL KALCLR( XKGS, PGS )
XKGS(1) = 0.25S*GM
XKGS(2) = GDH
PGS(1) = 0.1090688
PGS(6) = 0.01378051757
PGS(11) = 0.1090688
PGS(16) = 0.0001355074364
QGS(2) = 0.0002328737791
QGS(3) = 0.0006379424
QGS(4) = 0.002524766387

```

```

C GS.COMMON
301 CONTINUE

```

```

A15 = 0.90484S
IF ( GSCORT ) A15 = 0.99335S
TF15 = A15*(TF15-TM)+TM
TF15C = TM - TF15
UF1 = 0.98420S*( UF1 - U0 ) + U0
UF1C = U0 - UF1
UF2 = 0.98420S*( UF2 - UF1C ) + UF1C
SH0 = SH
SH = 0.9S*SH + 0.00066S*VHTN*COSX( ATGSN ) +
$ 0.04584S*Z/( 0.64S - 0.36S*GH )
IF ( 0.0S .GT. SH ) SH = 0.0S
TF1 = 0.90484S*( TF1 - TM ) + TM
TDM = TM - TF1
ATGSE = ATGSN + 0.1S*TDM
ATSW = ATSW - SH + SH0 = 0.00085S*VHTN*COSX( ATGSE )
IF ( IATSW .EQ. 1 ) ATSW = 0.0S
VHTN = U0 + ATSW
FK = -VHTN*TDM/( 0.17338S*SH )
H = 1.0
ZK = 0.25S*GM
IF ( GSIT .NE. 0 )
$ CALL KFIL2( TRGS, XKGS, H, ZK, FK, SDUMG, CDUMG )
GSIT = 1
GH = XKGS(1)
GH = GH/0.25S
GDH = XKGS(2)
GDF = 0.9S*( GDF - GDH ) + GDH
ATGSN = -0.60952S*GDF*SH/VHTN

```

```

C TDCA.SUB
TDCAV = ( 0.99999S - VHTN/0.42857S )*GH/0.37765S
TDCAV = TDCAV + ( 0.88328S*GDH + 0.44164S*GDF )/VHTN
TDCA = TDCAV*SH - TF15C

```

```

GO TO ( 310, 315, 320, 350 ), GSSW

```

```

C PRECAP CAP TRK IDLE

```

```

C GS.PRECAP.COMP

```

```

310 CONTINUE

```

```

C   GS.P.TO.CAP
    IF ( SENSW(5) ) GO TO 311
    IF ( ( GH ) .GT. 0.116155 ) GO TO 350
311  CONTINUE
    GSSW = 2
    GSCORT = .TRUE.
    TC = TM

C   GS.CAP.COMP
315  CONTINUE
    RPN = SH*GH
    RPN = SSIGN( RPN, ATGSN )
    TDC = -0.023145*VHTN*( 0.5S - 0.5S*COSX( ATGSN ) ) /RPN
    TC = TC + TDC
    HPTR = ( TC - TM )/0.33333S - UF2/0.18700S

C   GS.C.TO.TRK
    IF ( GH .GT. 0.08138S ) GO TO 325
    GSSW = 3
    TDCAT = HPTR - TDCA

C   GS.TRACK.COMP
320  CONTINUE
    HPTR = TDCA + TDCAT
    TDCAT = 0.985S*TDCAT

325  CONTINUE
    AED1 = -0.18S*HPTR
C   THE = SIGN TAKES CARE OF THE DAC INVERSION
    CALL QWJDAS( AED1, 7, IERR )
C   MAG. OF AED1 = ATSW = 100.0 KNOTS
    AED1 = ATSW/0.11905S
    CALL QWJDAS( AED1, 2, IERR )
    AED1 = ( GH - GM )/0.15S
    CALL QWJDAS( AED1, 6, IERR )
    IF ( GSSW .EQ. 4 ) GO TO 326
C   ONLY FOR HYBRID ALT. HOLD MODE
    CALL KFILL( PGS, YRGS, GGS, 1.0, RG(GSSW), SDUMG, CDUMG )
326  CONTINUE

    IF ( ITAPE .EQ. 1 ) CALL QMON( 16, TM, QL )
    IF ( ITAPE .EQ. 2 ) CALL QMON( 16, QL, DUMMY )

    ICOUNT = ICOUNT + 1

    GO TO 9
C   GO TO WAIT LOOP
C   ***   END OF ILS PROGRAM LOOP

C XXXXX THE FOLLOWING GS ALT. HOLD MODE DOES NOT EXIST IN REAL PGM
350  CONTINUE
    IF ( SENSW(5) .AND. NARLGS .AND. GSSW .EQ. 4 ) GO TO 300
    TMF1 = 0.99335S*( TMF1 - TM ) + TM
    TMF1C = TM - TMF1
    HF15 = 0.99335S*( HF15 - Z ) + Z
    HF15C = Z - HF15
    HPTR = AH - Z - TMF1C - HF15C - 0.2*ADV(10)

```


AD-A041 626 ROCKWELL INTERNATIONAL CEDAR RAPIDS IOWA COLLINS RA--ETC F/G 17/7
DIGITAL FLIGHT DIRECTOR COMPUTER.(U)
APR 75 J P DESMOND, G E FORQUER

UNCLASSIFIED

ASD-TR-76-3

F33657-73-C-0120
NL

4 OF 4
ADA041626



END

DATE
FILMED
8-77

GO TO 325

```
400 CONTINUE
C TAC.INIT
  IF ( .NOT. FTIME ) GO TO 401
  CALL LATIC
  PSINOW = GAMH
  PL(1) = 0.00297423164
  PL(6) = 0.9536825019
  PL(11) = 1.0
  PL(16) = 0.0004274698477
  QL02 = 0.8919864872
  QL(3) = 0.0003169697243
  QL(4) = 0.004067843615
  TRLAT(3) = 0.0002222222222
  LC = 0.05
  RL = 0.0305175

C TAC.COMMON
401 CONTINUE
  IF ( IT .GT. 0 ) GO TO 402
  XKL(1) = QM*RH
  XKL(2) = -U0*SINX( PSIECR )/0.957449
  X2L = XKL(2)
  XSIL = GAMH

402 CONTINUE
  CALL LATCOM(0.9999999,RH,QM,-0.083338,0.000238,0.999998,0.204538)
  GP = 0.9999999
  IF ( RH .GT. 0.097668 ) GP = 0.097668/RH
  QVSTP = 0.032815*U0
  X2L = 0.980155*( X2L - QDB ) + QDB
  XSIL = 0.980155*( XSIL - GAMH ) + GAMH
  YDT = X2L - VH*G*( GAMH - XSIL )/0.304768
  AED1 = XKL(1)
  PHICT = GP*( AED1 + 0.1926*YDT )/0.35815

  GO TO ( 410, 415, 420, 425 ), LATSW

C TACMH.COMP
410 CONTINUE
  AED1 = HH - PSIECR
  PHIC = XMANHO( AED1, U0 )

C TACMH.TO.CAP
  IF ( SENS(4) ) GO TO 411
C IF DISCRETE.INPUT.WORD .A. LBS0,MASK NEQ 0 THEN GOTO TRU S,
  IF ( (RH*SABS(QM) .GT. 0.099478 )
  $ .AND. ( IT .LT. 1 ) ) GO TO 430
  IT = IT + 1
  IF ( IT .LT. 30 ) GO TO 430
  ADJPHL = 0.359 +SLIMIT( 0.155, ( RH = 0.136725 )/0.520838 )
  IF ( ( ( PHICL*PHICT .LE. 0.05 ) .OR. ( SABS( PHICT ) .LE.
  $ SABS( PHICL*ADJPHL ) ) ) .OR.
  $ ( ( SABS( QH ) .LE. 0.088895 ) .AND.
  $ ( SABS( PHICT ) .GT. 0.005565 ) ) ) GO TO 430
411 CONTINUE
```

```

      LATSW = 2

C   TACAPTURE.COMP
415  CONTINUE
      RL = 0.0305175
      PSINOW = PSIECR
      IF ( ( SABS( PHICL ) .LE. SABS( PHICT ) ) .AND.
        * ( QH*PHICT .LT. 0.05 ) ) GO TO 416
      PHIC = PHICT
C   ITEST = .TRUE.
C   TACAP.TO.TRK
      IF ( SABS(QH) .GT. 0.029185 ) GO TO 430
      LATSW = 3
      GO TO 420
416  CONTINUE
      PHIC = PHICL
C   ITEST = .FALSE.
      GO TO 430

C   TAC.TRACK.COMP
420  CONTINUE
      PHIC = PHICT
      XRH = 0.039063*RH
      RL = LIMIT( 0.0076290395, XRH )
      PSINOW = 0.904845*( PSINOW = PSIECR ) + PSIECR
C   TTRK.TO.OVS
      IF ( RH .GT. OVSTP ) GO TO 430
      LATSW = 4

C   OVS.TO.CAP
425  CONTINUE
      AED1 = HH = PSIEMH
      PHIC = XMANHD( AED1, UO )
      IF ( RH .LE. OVSTP ) GO TO 430
C   *** MUST RE-IC
      LATSW = 2
      PL(1) = 0.00297423164
      PL(6) = 0.9536825019
      PL(11) = 1.0
      PL(16) = 0.0004274698477
      XKL(1) = QM*RH
      XKL(2) = -UO*SINX( PSIECR + PSINOW )/0.957445
      X2L = YKL(2)
      XKL(3) = 0.0
      XKL(4) = 0.0
      CTW = 0.05
      GAMH = PSIECR
      XSIL = PSIECR

430  CONTINUE
      DRH = RH
      CALL LATFIN( DRH )

      CALL QWJDAS( -QH, 0, IERR )
      AED1 = CTW/0.119055
      CALL QWJDAS( AED1, 1, IERR )
      YXXX = LATSW

```



```

AED1 = 0.1*XXXX
CALL QWJDAS( AED1, 2, IERR )
AED1 = ( QH - QM ) / 0.308
CALL QWJDAS( AED1, 3, IERR )
CALL QWJDAS( -QDB, 4, IERR )

IF ( ITAPE .EQ. 1 ) CALL QMON( 16, TM, QL )
IF ( ITAPE .EQ. 2 ) CALL QMON( 16, QL, DUMMY )

```

```

ICOUNT = ICOUNT + 1

```

```

GO TO 9

```

```

C GO TO WAIT LOOP

```

```

C *** END OF TACAN PROGRAM LOOP

```

```

000 CONTINUE

```

```

CALL QSIC( IERR )

```

```

C

```

```

PRINT DATA

```

```

CALL FPRT(XKL,4,8H XKL )

```

```

CALL FPRT(PL,16,8H PL )

```

```

CALL FPRT(QL,4,8H QL )

```

```

IF ( MODE .EQ. 3 .OR. GSSW .EQ. 4 ) GO TO 501

```

```

CALL FPRT(YKGS,4,8H YKGS )

```

```

CALL FPRT(PGS,16,8H PGS )

```

```

501 CONTINUE

```

```

TYPE 911, ICOUNT

```

```

IF ( ITAPE .EQ. 0 ) GO TO 1

```

```

CALL QMON(24,'16)

```

```

550 CONTINUE

```

```

CALL QMON(21,ADNA(1),1,'16)

```

```

TYPE 906

```

```

ACCEPT 901, MODE

```

```

GO TO ( 555, 560, 1 ), MODE

```

```

555 CONTINUE

```

```

DO 557 I = 1, ICOUNT

```

```

IF ( SENSW(1) ) GO TO 550

```

```

IF ( ITAPE .EQ. 2 ) GO TO 556

```

```

CALL QMON( 17, TM, QL )

```

```

TYPE 905, QM, QH, QDB, RA, RH, RHILS, PSIECR, GAMH, UO, VHG

```

```

GO TO 557

```

```

556 CONTINUE

```

```

CALL QMON( 17, QL, DUMMY )

```

```

CALL FPRT( XKL, 4, 8H XKL )

```

```

CALL FPRT( PL, 16, 8H PL )

```

```

CALL FPRT( CDUM, 4, 8H CDUM )

```

```

557 CONTINUE

```

```

GO TO 550

```

```

560 CONTINUE
TYPE 903
ACCEPT 904, IT1, IT2, IT3, IT4, IT5, IT6, IT7, IT8, IT9, IT10
DO 561 I = 1, ICOUNT
IF ( SENSW(1) ) GO TO 550
CALL QMON(17, LOK(1), LOK(140) )
TYPE 905, LOK(IT1), LOK(IT2), LOK(IT3), LOK(IT4), LOK(IT5),
$      LOK(IT6), LOK(IT7), LOK(IT8), LOK(IT9), LOK(IT10)
561 CONTINUE
GO TO 550

```

```

900 FORMAT( 43H = 1, 2, OR 3 FOR HDG, ILS OR TACAN, RESP.)
901 FORMAT(I60)
902 FORMAT(/, 2E15.8, /)
903 FORMAT( 51H = 10-3 DIGIT INTEGERS)
904 FORMAT( 10I3 )
905 FORMAT( 10S7 )
906 FORMAT( 8H = MODE)
907 FORMAT( 15H LONG. AXIS DATA)
908 FORMAT( 14H LAT. AXIS DATA)
909 FORMAT( 4(S7, 2X) )
910 FORMAT( 47H = 0, 1, OR 2 = NO TAPE, SF, OR FT DAT, RESPT.)
911 FORMAT(10H ICOUNT = , I5 )
912 FORMAT( 7H = INF)
913 FORMAT( 8H = ICTW)
914 FORMAT( 7H = IRH)
920 FORMAT( 9H = IATSW)
923 FORMAT( F20.10)
932 FORMAT( 49H = 1, 2, 3, 4, OR 5 FOR L, LK, G, GK, OR GENERAL)
END
C SUB1

```

```

SUBROUTINE LATCOM( RH1, RH2, SQM, KFK, KCTE, KSCAL1, KSCAL2 )
C LAT.COMMON
DIMENSION QL(4), XKL(4), TRLAT(4), CDUM(4), PL(16)
REAL SDUM, SDUMG, FK, ZK, QL02, DRH, RL
LOGICAL FTIME, SENSW
INTEGER IT, LATSW
SCALED FRACTION SABS, SSORT, SSIGN, SINX, COSX, ATANB, SLIMIT
SCALED FRACTION RHILS
SCALED FRACTION TH, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
$ SINPHI, COSPHI, RA, RE, RD, AY, RH, KFK, QH0, QH, QDB, GAME,
$ CTE, KCTE, CTW, WH, QH, CWH, SWH, AED1, VH, GAMH, PHICL, KY,
$ PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XSIL,
$ KSCAL1, KSCAL2, LC, PHILAG, AED2, RH1, RH2, SQM,
$ YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
$ TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
$ HPTR, TDCA, TDCAT
COMMON TH, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
$ RA, RE, RD, AY, RH, QH0, QH, QDB, GAME,
$ CTE, CTW, WH, QH, CWH, SWH, AED1, VH, GAMH, PHICL, KY,
$ PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XSIL,
$ RHILS,

```

```

*   QL, FK, RRH, QL02, TRLAT, XKL, SDUM, CDUM, PL, RL, IT,
*   DUMMY,
*   YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
*   TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
*   HPTR, TDCAT, TDCAT, LC,
*   LATSW, FTIME,
*   XXXXX, ICTW
SINPHI = SINX(PHI)
COSPHI = COSX(PHI)
QL(2) = QL02
  IF ( IT .GT. 25 ) QL(2) = 0.00025 * QL02
      DRH = RH2
      ZK = SQM
      FK = KFK*SINPHI*COSX( GAMH )/( RH1*COSPHI )
  IF ( IT .GT. 0 ) CALL KFIL2( TRLAT, XKL, DRH, ZK, FK, SDUM, CDUM )
QH0 = QH
QH = XKL(1)
  QH = QH/RH2
QDR = XKL(2)
GAME = GAMH + 0.00080S*SINPHI/( VH*G*COSPHI )
CTE = RH*( QH0 - QH ) - KCTE*VH*G*SINX( GAME )
CTE = SLIMIT( 0.02572S, CTE )
CTW = CTW + CTE/LC
  IF ( ICTW .EQ. 1 ) CTW = 0.0S
WH = SABS( CTW )
OH = SSIGN( 0.5S, CTW )
CWH = WH*COSX( OH - PSIECR )
SWH = WH*SINX( OH - PSIECR )
AED1 = UO + CWH
VHG = SSQRT( SWH*SWH + AED1*AED1 )
GAMH = PSIECR + ATAN8( SWH, AED1 )
AED1 = KSCAL2*VHG*VHG*( 0.49999S - 0.49999S*COSX( GAMH ) )
AED1 = -SSIGN( AED1, GAMH )
AED2 = KSCAL1*RH*SABS( OH )
PHICL = ATAN8( AED1, AED2 )
RETURN
END
C   SUB2

```

```

SUBROUTINE LATFIN( XRM )
C   LAT.FINISH.COMMON
  DIMENSION QL(4), XKL(4), TRLAT(4), CDUM(4), PL(16)
  REAL SDUM, SDUMG, FK, ZK, RL, XRM
  LOGICAL FTIME
  SCALED FRACTION SLIMIT
  SCALED FRACTION RHLS
  SCALED FRACTION TM, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
S   SINPHI, COSPHI, RA, RE, RD, AY, RH, KFK, QH0, QH, QDR, GAME,
S   CTE, KCTE, CTW, WH, OH, CWH, SWH, AED1, VHG, GAMH, PHICL, KY,
S   PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
S   KSCAL1, KSCAL2, LC, PHILAG,
S   YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
S   TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
S   HPTR, TDCAT, TDCAT, XRM
  COMMON          TM, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
S   RA, RE, RD, AY, RH, QH0, QH, QDR, GAME,
S   CTE, CTW, WH, OH, CWH, SWH, AED1, VHG, GAMH, PHICL, KY,

```



```

S  PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
S  RHILS,
S  GL, FK, RRH, QL02, TRLAT, XKL, SDUM, CDUM, PL, RL, IT,
S  DUMMY,
S  YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
S  TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
S  HPTP, TDCA, TDCAT, LC,
S  LATSW, FTIME,
S  XXXXX, ICTW
C XX PHILAG
  PHII = PHIC - PHOLD
  PHIC = PHOLD + SLIMIT( 0.002775, PHII )
  PHIC = SLIMIT( 0.16875, PHIC )
  PHOLD = PHIC
  VPTR = ( PHIC - PHII ) / (-0.555565)
C    THE - SIGN TAKES CARE OF THE DAC INVERSION
  CALL GWJDAS( VPTR, 5, IERR )
  IF ( IT .GT. 0 ) CALL KFIL1( PL, TRLAT, GL, XRH, RL, SDUM, CDUM )
  RETURN
END
C  SUB3

SUBROUTINE KFIL1( P, TRANS, Q, H, R, SDUM, CDUM )
  DIMENSION P(16), PP(16), TRANS(4), Q(4), CDUM(4), DUM(16)
  CALL KMATRIX( DUM, TRANS, P )
  CALL KMATRIX( PP, TRANS, DUM )
  PP(6) = PP(6) + Q(2)
  PP(11) = PP(11) + Q(3)
  PP(16) = PP(16) + Q(4)
  DO 1 I = 1,4
    J = 4*(I-1)+1
    K = 4*(I-1)+3
  1   CDUM(I) = PP(J) + H*PP(K)
  SDUM = CDUM(1) + CDUM(3)*H + R
  DO 2 I = 1,4
    DO 2 J = 1,4
      K = 4*(I-1)+J
      L = 8+J
  2   P(K) = PP(K) - CDUM(I)*( PP(J) + PP(L)*H )/SDUM
  RETURN
END
C  SUB4

SUBROUTINE KFIL2( TRANS, XK, H, ZK, FK, SDUM, CDUM )
  DIMENSION TRANS(4), CDUM(4), XKP(4), XK(4)
  XKP(1) = XK(1) + TRANS(3)*XK(2)
  XKP(2) = XK(2) + 0.02844444444*( XK(4) + FK )
  XKP(3) = TRANS(1)*XK(3)
  XKP(4) = TRANS(2)*XK(4)
  SDUM3 = H*( ZK - XKP(3) ) - XKP(1)
  DO 1 I = 1,4
  1   XK(I) = XKP(I) + CDUM(I)*SDUM3/SDUM
  RETURN
END
C  SUB5

SUBROUTINE KMATRIX( RES, TR, M )

```

```

REAL M
DIMENSION RES(16), TR(4), M(16)
RES( 1) = M( 1) + TR(3)*M( 5)
RES( 5) = M( 2) + TR(3)*M( 6)
RES( 9) = M( 3) + TR(3)*M( 7)
RES(13) = M( 4) + TR(3)*M( 8)
RES( 2) = M( 5) + 0.02844444444444444*M(13)
RES( 6) = M( 6) + 0.02844444444444444*M(14)
RES(10) = M( 7) + 0.02844444444444444*M(15)
RES(14) = M( 8) + 0.02844444444444444*M(16)
RES( 3) = TR( 1) * M( 9)
RES( 7) = TR( 1) * M(10)
RES(11) = TR( 1) * M(11)
RES(15) = TR( 1) * M(12)
RES( 4) = TR( 2) * M(13)
RES( 8) = TR( 2) * M(14)
RES(12) = TR( 2) * M(15)
RES(16) = TR( 2) * M(16)
RETURN
END
C SUB6

```

```

SUBROUTINE LATIC
C LAT,INIT
DIMENSION QL(4), XKL(4), TRLAT(4), CDUM(4), PL(16)
REAL SDUM, SDUMG, FK, ZK, RL
LOGICAL FTIME
SCALED FRACTION RHILS
SCALED FRACTION TM, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
S SINPHI, COSPHI, RA, RE, RD, AY, RH, KFK, QH0, QH, QDB, GAME,
S CTE, KCTE, CTW, WH, OH, CWH, SWH, AED1, VHG, GAMH, PHICL, KY,
S PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
S KSCAL1, KSCAL2, LC, PHILAG,
S YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
S TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
S HPTR, TDCA, TDCAT
COMMON TM, PHI, DMER, PSIEMH, PSIECR, QM, TQM, GM, UO, Z,
S RA, RE, RD, AY, RH, QH0, QH, QDB, GAME,
S CTE, CTW, WH, OH, CWH, SWH, AED1, VHG, GAMH, PHICL, KY,
S PHICT, PHIC, HPHIC, PHII, PHOLD, VPTR, GP, OVSTP, X2L, XS1L,
S RHILS,
S QL, FK, RRH, QL02, TRLAT, XKL, SDUM, CDUM, PL, RL, IT,
S DUMMY,
S YDT, ADJPHL, PSINOW, A15, TF15, TF15C, UF1, UF1C, UF2, SH0, SH,
S TF1, TDM, ATGSE, ATSW, VHTN, GH, GDH, GDF, ATGSN, RPN, TDC, TC,
S HPTR, TDCA, TDCAT, LC,
S LATSW, FTIME,
S XXXXX, ICTW
CALL KALCLR( XKL, PL )
RL = 0.0
IT = 0
QM = QM
CTW = 0.05
VHG = UO
GAMH = PSIECR
PHOLD = PHI
PHILAG = 0.05

```

```

RETURN
END
C SUB7

SUBROUTINE KALCLR( XK, P )
DIMENSION XK(4), P(16)
DO 1 I = 1,4
  XK(I) = 0.0
DO 1 J = 1,4
  K = 4*(I-1)+J
1 P(K) = 0.0
RETURN
END
C SUB8

SCALED FRACTION FUNCTION XMANHD( X, UO )
SCALED FRACTION X, UO, GHH, SLIMIT, AED1
  GHH = ( UO - 0.35743S )/0.53572S
  GHH = SLIMIT( 0.33333S, GHH ) + 0.66666S
  AED1 = SLIMIT( 0.16666S, X )
  XMANHD = GHH*AED1/0.33333S
RETURN
END
C SUB9

SCALED FRACTION FUNCTION SINX( Y )
SCALED FRACTION X, Y, A0, A1, A2, A3, A4, SABS, SSIGN, X2, AED1
DATA A0, A1, A2, A3, A4/ 0.57079S, -0.64596S, 0.79688S,
$ -0.46722S, 0.01508S /
X = Y
C ONLY NEED FOR HYBRID
IF ( 0.5S .GT. SABS( X ) ) GO TO 2
IF ( X .EQ. -0.5S ) GO TO 1
X = SSIGN( 0.99999S-SABS(X), X )
IF ( X .EQ. +0.99999S ) GO TO 2
1 X = 0.99998S*X
2 X = X + X
X2 = X*X
AED1 = ( A3 + A4*X2 )*X2
AED1 = ( A2 + 0.1S*AED1 )*X2
AED1 = ( A1 + 0.1S*AED1 )*X2
SINX = X*( A0 + AED1 ) + X
RETURN
END
C SUB10

SCALED FRACTION FUNCTION COSX( Y )
SCALED FRACTION X, Y, SINX
X = Y
C ONLY NEED FOR HYBRID
X = 0.5S + X
COSX = SINX( X )
RETURN
END
C SUB11

```



```

SCALED FRACTION FUNCTION ATAN8( Y, X )
SCALED FRACTION X, Y, SABS, ANSWER, Z, ZZ, A1, A2, A3, B0, B1, B2, B3
DATA A1, A2, A3, B0, B1, B2, B3 / 0.147598, 0.110438, 0.004148,
$      0.055598, 0.845278, 0.414548, 0.181088 /

```

```

IF ( SABS( X ) .NE. SABS( Y ) ) GO TO 1

```

```

ANSWER = 0.255

```

```

GO TO 2

```

```

1 CONTINUE

```

```

Z = X/Y

```

```

IF ( SABS( X ) .GT. SABS( Y ) ) Z = Y/X

```

```

Z = SABS(Z)

```

```

ZZ = 0.125*Z*Z

```

```

ANSWER = Z*( B0 + A1/( ZZ + B1 - A2/( ZZ + B2 - A3/(ZZ+B3))))

```

```

IF ( SABS( Y ) .GT. SABS( X ) ) ANSWER = 0.55 - ANSWER

```

```

2 CONTINUE

```

```

IF ( X .GE. 0.05 ) GO TO 3

```

```

ANSWER = 0.999998 - ANSWER

```

```

3 CONTINUE

```

```

IF ( Y .LT. 0.05 ) ANSWER = - ANSWER

```

```

ATAN8 = ANSWER

```

```

RETURN

```

```

END

```

```

C SUB12

```

```

SCALED FRACTION FUNCTION SLIMIT( L, VAL )

```

```

SCALED FRACTION L, VAL

```

```

SCALED FRACTION SABS, SSIGN

```

```

SLIMIT = VAL

```

```

IF ( SABS( VAL ) .GT. L ) SLIMIT = SSIGN( L, VAL )

```

```

RETURN

```

```

END

```

```

C SUB13

```

```

REAL FUNCTION LIMIT( L, VAL )

```

```

REAL L

```

```

LIMIT = VAL

```

```

IF ( ABS( VAL ) .GT. L ) LIMIT = SIGN( L, VAL )

```

```

RETURN

```

```

END

```

```

C SUB14

```

```

SUBROUTINE FPRT( A, NE, TITLE )

```

```

DIMENSION A(16), TITLE(2)

```

```

TYPE 100, TITLE(1), TITLE(2)

```

```

DO 1 I = 1, NE, 4

```

```

TYPE 101, A(I), A(I+1), A(I+2), A(I+3)

```

```

1 CONTINUE

```

```

100 FORMAT(/,5X,2A4)

```

```

101 FORMAT(4E14.8)

```

```

RETURN

```

```

END

```

```

C SUB15

```

```

SUBROUTINE SPRT( A, NE, TITLE )

```

```

DIMENSION TITLE(2)

```

```

SCALED FRACTION A(4)

```

```

TYPE 100, TITLE(1), TITLE(2)

```

```
      TYPE 101, A(1), A(2), A(3), A(4)
100  FORMAT(/,5X,2A4)
101  FORMAT(4(S7,2X))
      RETURN
      END
```

```
*
#W,15
#W,16
#L,CIG,14
#R,70000,101000,15000
#C,15
#N,SFDFDP,15,2
#G,67300
#L,OBJTAA,16↑
#L,OMON,14↑
#L,LINKN,14↑
#L,RTL,14↑
#F
#L,ROED,14↑
#UL
#EL
#M
#I,2
```

APPENDIX E

AED PROGRAM LISTING (SCALED)

THIS APPENDIX PROVIDES THE DFDC "AED" SYSTEM SOFTWARE DESCRIPTION.
THIS "AED" SOFTWARE DESCRIPTION PROVIDES THE "COMPLETE" SOFTWARE
DOCUMENTATION FOR THE DFDC. THE VARIABLES CODED IN THIS SYSTEM
DESCRIPTION ARE DEFINED IN APPENDIX B. THE FLOW CHARTS FOR THE
PROGRAM ARE ILLUSTRATED IN APPENDIX A.

CUFN AED,* INTSRV,INTSRV
BEGIN

... DIGITAL FLIGHT DIRECTOR SYSTEM PROGRAM, AED DESCRIPTION //

```

... *****// PRESET SIMCOUNT = 0 $,
... *****// PROCEDURE FORINT, SIMAC $,
... *****// PROCEDURE ASMCT $,
... *****// REAL QH,WH,VHG,GAMH,GH,SH,ATSW,VHTN,ATGSN $,
... *****// INTEGER II,SIMCOUNT $,
... *****// INTEGER J,KOUNT $,
... *****// INTEGER PROCEDURE MAXCNT,KMAX $,
... *****// REAL ARRAY XX(100),CRSER(100),LOC DV(100),GSDV(100),
... *****// RANGE(100),HPNTR(100),VPNTR(100), PITCH(100),
... *****// BANK(100),ALT(100),RHOUT(100),QHOUT(100),WHOUT(100),
... *****// VHGOUT(100),GAMHO(100),QHERR(100),GHOUT(100),SHOUT(100),
... *****// ATSWO(100),VHTNO(100),ATGSNO(100) $,
... *****// PROCEDURE PLRF29,NEW.PAG $,
... *****// PRESET J = -1 $,
... *****// PRESET KOUNT = 1 $,
... *****// BOOLEAN PROCEDURE CHKPT $,
... *****// PROCEDURE CARET $,
... *****// PROCEDURE PRTA $,
... *****// DEFINE BOOLEAN PROCEDURE SAMPLE TOBE
... *****// BEGIN
... *****// FORINT(SIMAC,IO,MATRIX,15 .A. DISCRETE.INPUT,WORD) $,
... *****// BPPR(16) = IO,MATRIX(14C) $, ... BYPASS DFDPGM RANGE //
... *****// SIMCOUNT = SIMCOUNT + 1 $,
... *****// IF CKPT THEN BEGIN
... *****// FOR II=0 STEP 1 UNTIL 31 DO
... *****// PRTA(ASMFL0,0,14,IO,MATRIX(II)) $,
... *****// CARET() $,
... *****// END $,
... *****// IF KOUNT LES KMAX() THEN KOUNT = KOUNT+1
... *****// ELSE BEGIN
... *****// KOUNT = 1 $,
... *****// J = J+1 $,
... *****// XX(J) = J*KMAX()*0.05 $,
... *****// CRSER(J) = IO.MATRIX(20C) $,
... *****// LOC DV(J) = IO.MATRIX(22C) $,
... *****// GSDV(J) = IO.MATRIX(24C) $,
... *****// RANGE(J) = IO.MATRIX(14C) $,
... *****// HPNTR(J) = IO.MATRIX( 1C) $,
... *****// VPNTR(J) = IO.MATRIX( 0 ) $,
... *****// PITCH(J) = IO.MATRIX(30C) $,
... *****// BANK(J) = IO.MATRIX(31C) $,
... *****// ALT(J) = IO.MATRIX(27C) $,
... *****// RHOUT(J) = RH $,
... *****// QHOUT(J) = QH $,

```



```

... *****//      FORINT(PLRF29,XX,QHERR ,1,100,100) $,
... *****//      NEW,PAG() $,
... *****//      PRT(0,.C. /GH OUTPUT          /) $,
... *****//      FORINT(PLRF29,XX,GHOUT ,1,100,100) $,
... *****//      NEW,PAG() $,
... *****//      PRT(0,.C. /SH                      /) $,
... *****//      FORINT(PLRF29,XX,SHOUT ,1,100,100) $,
... *****//      NEW,PAG() $,
... *****//      PRT(0,.C. /ATSW                     /) $,
... *****//      FORINT(PLRF29,XX,ATSWO ,1,100,100) $,
... *****//      NEW,PAG() $,
... *****//      PRT(0,.C. /VHTN                      /) $,
... *****//      FORINT(PLRF29,XX,VHTNO ,1,100,100) $,
... *****//      NEW,PAG() $,
... *****//      PRT(0,.C. /ATGSN                     /) $,
... *****//      FORINT(PLRF29,XX,ATGSNO,1,100,100) $,
... *****//      END $,
... *****//      CKPT = CHKPT(SIMCOUNT) $,
... *****//      END $,

... *****//      BOOLEAN ARRAY BOOLOUT(15) $,

      DEFINE PROCEDURE DIG.OUT(BIT,NO,BVALU ) WHERE
      INTEGER BIT,NO $, BOOLEAN BVALU TOBE
... *****//      BOOLOUT(BIT,NO) = BVALU $,

... *****//      DEFINE PROCEDURE DWORD.OUT(I) WHERE INTEGER I TOBE
... *****//      BEGIN
... *****//      INTEGER L $,
... *****//      FOR L=1 STEP 1 UNTIL 16 DO
... *****//      BOOLOUT(L-1) = ((1 .LS. (L-1)) .A. I) NEG 0 $,
... *****//      END $,

... *****//      REAL ARRAY IO.MATRIX(31) $,

... *****//      DEFINE PROCEDURE OUTPUT(ADDRESS,RVALU)
... *****//      WHERE INTEGER ADDRESS $, REAL RVALU TOBE
... *****//      IO.MATRIX(ADDRESS) = RVALU $,

... *****//      DEFINE PROCEDURE IN.REQUEST(ADDR) WHERE INTEGER ADDR
... *****//      TOBE WHOLE(INP.STORAGE)=IO.MATRIX(ADDR) $,

... ADD      DEFINE PROCEDURE IN.REQUEST(ADDR) WHERE INTEGER ADDR
... ADD      DUMP = WHOLE(WHOLE(LIST.ADDRESS)) $,
... ADD COMMENT      DUMP CLEANS NOISE OFF STACK,ALSO NOTE THAT
      COMPONENT TYPE OF INNER WHOLE IS INCONSISTENT //

... *****//
... GLOBAL VARIABLES (COMMON TO ALL 3 STACKS) //
      REAL PSIEHDOG,PSIECRS,KSINPHI,KCOSPHI,QM,HSINR,HCOSR,TSINR,

```



```

    USINR,HCOSR,
    TCOSR,SINPHI,COSPHI,RH,UO,TM,SDUM,SDUMG,PHI $,

    REAL QLO2 $,

    COMMON GSSTACK,LATSTACK,INTSTACK,DISCRETE,INPUT,WORD,FIRSTIME,
    BFFR,PPGS,PGS,QGS,XKGS,
    PPL,PL,QL,XKL,XKPL,CDUM,CDUMG,GSCORT,
    SINPHI,COSPHI,RH,TM,SDUM,SDUMG,MODE
... *****// ,CKPT,QH,WH,VHG,GAMH,GH, ATSW,VHTN,ATGSN ,PHI,NABLGS
$,

    REAL ARRAY GSSTACK(99),LATSTACK(99)INTSTACK(99) $,
    REAL ARRAY PPGS(16),PGS(16),QGS(4),XKGS(4),
    PPL(16), PL(16), QL(4), XKL(4) $,
    REAL ARRAY TRLAT(3),TRGS(3),CDUM(4),CDUMG(4) $,
    BOOLEAN GSCORT, FIRSTIME $,
    INTEGER DISCRETE,INPUT,WORD $,
    INTEGER MODE $,
    POINTER IN, PTR $,
... *****// BOOLEAN NABLGS $,
    PPSET TRLAT(1) = 0.9753099 $,
    PPSET TRLAT(2) = 0.9048374 $,

... THE FOLLOWING VARIABLES SHARE SPACE IN BFFR //
... UO = BFFR(0),PSIEMHOG=BFFR(1), .... TSINR=BFFR(8),TCOSR=BFFR(9) //

    SYNONYMS BEGIN
        PSIEMHOG = GM $,
        PSIECPS = Z $,
        KCOSPHI = KCOSOMEGA $,
        KSINPHI = KSINOMEGA $,
        QM = STC1, TOM $,
        HSINR = HPTR.INP $,
        HCOSR = VPTR.INP $,
    END $,

    REAL COMPONENT WHOLE $,
    WHOLE $$$ 0 $,
    PROCEDURE PTR,FLAG
    REAL PROCEDURE KALCLR,KFIL1,KFIL2 $,
    REAL PROCEDURE SQRT, EXP, ATAN8, ATAN, SIN, COS $,
    REAL PROCEDURE LIMIT,SIGN $,
    PROCEDURE FAILURE,WARNING $,
... *****// BOOLEAN CKPT $,
... *****// PROCEDURE ASHDEC $,
... *****// PROCEDURE ASMFLO $,
... *****// PROCEDURE PRT $,

... *****//

... LOCAL VARIABLES (LOCAL TO THIS STACK ONLY) //

    INTEGER RCOP,RCOPU $,

```

```

INTEGER DIS.WORD,IC $,
INTEGER MANTAS.MASK $,
INTEGER MODELA $,
INTEGER TAC.MASK,MH.MASK,ILS.MASK,ST.MASK,N,LIST,ADDRESS $,
BOOLEAN DONE $,
BOOLEAN ASREF $,
REAL ASR $,
REAL ANGH,ANGT,ANGU,HDIGIT,TDIGIT,UDIGIT $,
REAL ROLD,XSTART,XRG,VAVG,SIGNN $,
REAL RANGEQ,RANGET,VERR,RCOPUT,RUPT $,
REAL RANGE $,
REAL ARRAY BFER(11),IRFER(11) $,
INTEGER ARRAY ILS.LIST(4),TAC.LIST(10),ST.LIST(6),
ILSA.LIST(5) $,
... ****// PROCEDURE EXIT $,
    POINTER HDG.ERROR,CRS.ERROR,ROLL.SIN,ROLL.COS,LOC.DEV,
    STEST1,STEST2,H.PTR,V.PTR,TAC.DEV,H.SINR,H.COSR,T.SINR,
    T.COSR,GS.DEV,RADAR.ALT,P.SIN,P.COS,INP.DISCRETES,
... I/O //
    U.SINR,U.COSR,AS.PEF,
    INP.STORAGE,OUT.DISCRETES,OUT.VPTR,OUT.HPTR,MSK.REG,
    COMP.AS,MAN.AS $,
... *****// INTEGER PROCEDURE SET.INPUT $,

```

```

SYNONYMS BEGIN 4 = TAC.MASK $,
                1 = MH.MASK $,
                2 = ILS.MASK $,
                8 = ST.MASK $,
END $,

```

```

SYNONYMS BEGIN
21C = HDG.ERROR $,
20C = CRS.ERROR $,
12C = ROLL.SIN $,
13C = ROLL.COS $,
22C = LOC.DEV $,
32C = STEST1 $,
33C = U.SINR $,
35C = U.COSR $,
34C = AS.REF $,
30C = H.PTR $,
31C = V.PTR $,
23C = TAC.DEV $,
16C = H.SINR $,
17C = H.COSR $,
14C = T.SINR $,
15C = T.COSR $,
10C = P.SIN $,
11C = P.COS $,
24C = GS.DEV $,
27C = RADAR.ALT $,
36C = INP.DISCRETES $,
37C = INP.STORAGE $,
02C = OUT.DISCRETES $,
00C = OUT.VPTR $,
01C = OUT.HPTR $,
77C = MSK.REG $,
25C = COMP.AS $,

```

```

                260 = MAN,AS          $,
END $,
... NOTE THE ABOVE OCTAL INDICES WILL BE PREFACED WITH 177 //

```

```

    INTEGER ARRAY MSK,TBL(4),N,TBL(5),LST,AD,TBL(5) $,

```

```

PRESET DISCRETE,WORD,IC = 5400C $,
PRESET MSK,TBL(1) = TAC,MASK $,
PRESET MSK,TBL(2) = ST,MASK $,
PRESET MSK,TBL(3) = MH,MASK $,
PRESET MSK,TBL(4) = ILS,MASK $,
PRESET N,TBL(1) = 12 $,
PRESET N,TBL(2) = 9 $,
PRESET N,TBL(3) = 5 $,
PRESET N,TBL(4) = 6 $,
PRESET N,TBL(5) = 6 $,
PRESET LST,AD,TBL(1) = LOC TAC,LIST $,
PRESET LST,AD,TBL(2) = LOC ST,LIST $,
PRESET LST,AD,TBL(3) = LOC ILS,LIST $,
PRESET LST,AD,TBL(4) = LOC ILS,LIST $,
PRESET LST,AD,TBL(5) = LOC ILS,LIST $,

```

```

PRESET ILS,LIST(0) = HDG.ERROR $,
PRESET ILS,LIST(1) = CRS,ERROR $,
PRESET ILS,LIST(2) = ROLL,SIN $,
PRESET ILS,LIST(3) = ROLL,COS $,
PRESET ILS,LIST(4) = LOC,DEV $,
PRESET TAC,LIST(0) = HDG.ERROR $,
PRESET TAC,LIST(1) = CRS,ERROR $,
PRESET TAC,LIST(2) = ROLL,SIN $,
PRESET TAC,LIST(3) = ROLL,COS $,
PRESET TAC,LIST(4) = TAC,DEV $,
PRESET TAC,LIST(5) = H,SINR $,
PRESET TAC,LIST(6) = H,COSR $,
PRESET TAC,LIST(7) = T,SINR $,
PRESET TAC,LIST(8) = T,COSR $,
PRESET TAC,LIST(9) = U,SINR $,
PRESET TAC,LIST(10) = U,COSR $,
PRESET ST,LIST(0) = HDG.ERROR $,
PRESET ST,LIST(1) = CRS,ERROR $,
PRESET ST,LIST(2) = ROLL,SIN $,
PRESET ST,LIST(3) = ROLL,COS $,
PRESET ST,LIST(4) = STEST1 $,
PRESET ST,LIST(5) = H,PTR $,
PRESET ST,LIST(6) = V,PTR $,
PRESET ILSA,LIST(0) = GS,DEV $,
PRESET ILSA,LIST(1) = RADAR,ALT $,
PRESET ILSA,LIST(2) = P,SIN $,
PRESET ILSA,LIST(3) = P,COS $,

```

```

... *****//

```

```

DEFINE PROCEDURE MOVE(FROM,TU,K) WHERE REAL ARRAY FROM,TU $,
    INTEGER K TOBE BEGIN

```



```

        INTEGER I $,
        FOR I= K STEP -1 UNTIL 0 DO
            TU(I) = FROM(I) $,
        IN.PTR = LOC BFR $,
    END $,

```

```
POC $      MODELA = 0 $,
```

```
RSTART$
```

```

... ADD    SET.TIMER2.INTERRUPT.MASK() $,
... ADD    INITIALIZE.STACKS() $,
... ADD    FLCT = 0 $,
            GSCOPT = FALSE $,
            NARLGS = FALSE $,
            GS.INDEX = 9 $,
            DWORD.OUT(DISCRETE.WORD.IC) $,
            ... IC = GPDISENGAGE; HPTR, VPTR IN VIEW; FDVALID;
            ALL FLAGS OUT OF VIEW //
            DONE = TRUE $,
... ADD    GOTO WAITLP VIA INTRIN $,
... *****// WAITLCP $ IF SAMPLE() THEN EXIT() $,
... *****//     IF TIMER2 THEN TIME1.INTERRUPT.SERVICE()
... *****//     ELSE TIME2.INTERRUPT.SERVICE() $,

... *****//     BOOLEAN TIMER2 $,

... ****//     DEFINE PROCEDURE TIME1.INTERRUPT.SERVICE TOBE
TM1 $      BEGIN
... *****//     TIMER2 = FALSE $,
... *****//     IF CKPT THEN PRT(0,,C. /CHECKPOINT 3           //)$,
            INTEGER IM,ITBL $,
            ITBL = 5 $,
            FOR IM = 1 STEP 1 UNTIL 4 DO
                IF DISCRETE.INPUT.WORD .A. MSK.TBL(IM) NEQ 0
                    THEN ITBL = IM $,
            N = N.TBL(ITBL) $,
            LIST.ADDRESS = LST.AD.TBL(ITBL) $,
            MODE = ITBL $,
            IF (FIRSTIME = MODELA NEQ MODE)
                THEN BEGIN
                    MODELA = MODE $,
                    GOTO RSTART $,
                END $,
            GOTO INR $,
        END $,

```

```

... ****//     DEFINE PROCEDURE TIME2.INTERRUPT.SERVICE TOBE
TM2 $      BEGIN
            IF ( MODE EQ ILS.MODE ) THEN BEGIN
... *****//     TIMER2 = TRUE $,
... *****//     IF CKPT THEN PRT(0,,C. /CHECKPOINT 5
                    SET.APPR.MODE() $,
                    LIST.ADDRESS = LOC ILSA.LIST $,
                    N = 5 $,

```

```
INR $
```

```

... *****//      IN,PTR = LOC IRFFR $,
                     INT,NO = 2 $,
                     IF (DISCRETE.INPUT.WORD .A. MANTAS,MASK) NEQ 0
                     THEN BEGIN ASREF = TRUE $,
                                IN.REQUEST(AS,REF) $,
                                END
                     ELSE BEGIN      ASR = 1.0 $,
                                IN.REQUEST(COMP,AS) $,
                                END $,
                     IF CKPT THEN PRT(0,,C. /CHECKPOINT 4           /)$,
                     END $,
... ADD              GOTO INT.RET $,
... *****//      INPUT.INTERRUPT.SERVICE() $,
                     END $,

DEFINE PROCEDURE SET.APPR.MODE TOBE MODE = 6 $,

... *****//      DEFINE PROCEDURE INPUT.INTERRUPT.SERVICE TOBE
LP $      BEGIN
          IF ASREF THEN BEGIN
                      ASR = WHOLE(INP.STORAGE) $,
                      ASREF = FALSE $,
                      IN.REQUEST(MAN,AS) $,
... ADD              GOTO INT.RET $,
... *****//      GOTO LP $,
                      END $,
                      WHOLE(IN,PTR) = WHOLE(INP.STORAGE) $,
                      IN,PTR = IN,PTR + 1 $,
                      IF (N = N-1) NEQ 0 THEN BEGIN
... ADD              IN.REQUEST(LIST,ADDRESS=LIST,ADDRESS + 1) $,
... *****//      GOTO INT.RET $,
                      GOTO LP $,
                      END $,
                      IF DONE THEN DONE = FALSE ELSE FAILURE.WARNING() $,
                      MOVE(IRFFR,BFFR,11) $,
                      BFFR(0) = BFFR(0)/ASR ... DIVIDE BY AIRSPEED REFERENCE $,
                      TM = PHI = ATAN8(BFFR(3),BFFR(4)) $,
... *****//      GOTO RRG $,
... DME RANGE INPUT PRE-PROCESSING //
          IF MODE EQL 1 THEN
          BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /CHECKPOINT 2           /)$,
... WHEN MOD. DME RANGE UP-DATE THEN MAKE A/C CHANGE, SCALE //
... //      ANGH = ATAN8(BFFR(6),BFFR(7)) ... HSINR,HCOSR $,
          ANG.TO.DIGIT( ANGH, AED1 ) $,
          IF ( AED1 LES 0.55 ) THEN HDIGIT = AED1*1.95
          ELSE HDIGIT = 0.0 $,
... //      ANG1 = ATAN8(BFFR(8),BFFR(9)) ... TSINR,TCOSR $,
          ANG.TO.DIGIT(ANG1,AED1) $,
          TDIGIT = 0.195*AED1 $,
... //      ANGU = ATAN8(BFFR(10),BFFR(11)) ... USINR,UCOSR $,
          ANG.TO.DIGIT(ANGU,AED1) $,
          UDIGIT = 0.0195*AED1 $,

```

```
DEFINE PROCEDURE ANG.TO.DIGIT(ANG,DIGIT)
```

```
WHERE REAL DIGIT,ANG TOBE
```

```

BEGIN   ANG = ANG + 0.1 $,
DIGIT = IF ANG LES -0.8 THEN 0.5
        ELSE IF ANG LES -0.6 THEN 0.6
        ELSE IF ANG LES -0.4 THEN 0.7
        ELSE IF ANG LES -0.2 THEN 0.8
        ELSE IF ANG LES  0.0 THEN 0.9
        ELSE IF ANG LES  0.2 THEN 0.0
        ELSE IF ANG LES  0.4 THEN 0.1
        ELSE IF ANG LES  0.6 THEN 0.2
        ELSE IF ANG LES  0.8 THEN 0.3
        ELSE
                                0.4 $,
END $,
```

```
... TEST ONE RANGE CROSS OVER PROBLEM //
```

```
RCOP = RCOP + 1 $,
```

```
RCOPU = RCOPU + 1 $,
```

```
RCOPUT = TOIGIT + UDIGIT $,
```

```
IF (FIRSTIME) THEN
```

```
  BEGIN
```

```
    RUPT = RCOPUT $,
```

```
    RANGEQ = RCOPUT + HDIGIT $,
```

```
    RH = RANGEQ $,
```

```
    ROLD = RANGEQ $,
```

```
    SIGNN = 0.0 $,
```

```
  END $,
```

```
IF ((RCOPU GEQ 3) OR (ABS(RCOPUT-RUPT) LES 5.0/512.0)) THEN
```

```
  BEGIN
```

```
    RUPT = RCOPUT $,
```

```
    RCOPU = 0 $,
```

```
  END $,
```

```
RANGET = HDIGIT + RUPT $,
```

```
IF ((RCOP GEQ 3) OR (ABS(RANGET-RANGEQ) LES 50.0/512.0)) THEN
```

```
  BEGIN
```

```
    RANGEQ = RANGET $,
```

```
    RCOP = 0 $,
```

```
  END $,
```

```
... ONE RANGE LINEARIZATION //
```

```
... // AED1 = RANGEQ - ROLD $,
```

```
IF ( ( AED1 ) EQL 0.0 )
```

```
  THEN BEGIN
```

```
    RH = RH + SIGNN*ABS(VHG*COS(GAMH)) $,
```

```
  END
```

```
ELSE BEGIN
```

```
  SIGNN = SIGN(0.0233333333,AED1) $,
```

```
  RH = RANGEQ + 0.0009765625 * SIGN(0.0009765625,AED1) $,
```

```
  ROLD = RANGEQ $,
```

```
  END $,
```

```
END $,
```

```
... *****//   RNG $   RANGE = BFFR(8) $,   ... TSINR POSITION //
```

```
... *****// IF MODE EQL 1 THEN RH = RANGE $,
```

```
  PSIECRS = 0.5*PSIECRS ... FOR SCALED PROGRAM $,
```

```
  PSIEMHOG = 0.5*PSIEMHOG ... FOR SCALED PROGRAM $,
```



```

... ADD SELECT,STACK,DESCRIPTOR() $,
... ADD GOTO INT,RET $,
... *****// GOTO RETURN $,
      END $,

```

```

INT,RET $      INTERRUPT,RETURN() $,
      SWITCH INT,SW = TM1,TM2,LP,FLW,FLW,FLW,HLT,FLW $,
... NOTE IN ABOVE SW, 5 AND 6 ARE ILL.A AND ILL.OP,4 AND 7 UNUSED //
... *****// GOTO INT,SW(INT.NO + 1) $,
... ADD      INT.NO = T.O.S. //
... ADD      ITEMP1 = DISCRETE.INPUT,WORD //
... ADD      GOTO INT,SW(INT.NO) //

```

```

HLT $      STORE.TOP,WORD,TO(LOC GS,TOS) $,
      GOTO INT,RET $,

```

```

FLW $      FAILURE,WARNING() $,
      GOTO INT,RET $,

```

END FINI

SENTINEL CARD FOR U-1108 TERMINAL

```

* EDT,* LATPGM,LATPGM
  BEGIN

```

... THE FOLLOWING IS THE LATERAL PROGRAM BLOCK//

... *****//

```

... GLOBAL VARIABLES (COMMON TO ALL 3 STACKS) //
  REAL      PSTEMHDG,PSIECRS,KSINPHI,KCOSPHI,
      QM,HSINR,HCSR,TSINR,TCOSR,SINPHI,COSPHI,RH,UO,TM,SDUM,
      USINR,UCOSR,
      SDUMG,PHI $,

```

```

  COMMON GSSTACK,LATSTACK,INTSTACK,DISCRETE,INPUT,WORD,FIRSTIME,
      UO,PSTEMHDG,PSIECRS,KSINPHI,KCOSPHI,QM,HSINR,
      HCSR,TSINR,TCOSR,USINR,UCOSR,PPGS,PGS,QGS,XKGS,
      PPL,PL,QL,XKL,XKPL,CDUM,CDUMG,GSCORT,
      SINPHI,COSPHI,GLRL,RH,TM,SDUM,SDUMG,MODE

```

```

... *****// ,CKPT,QH,WH,VHG,GAMH,GH, ATSW,VHTN,ATGSN ,PHI,NABLGS
  $,

```

```

  REAL ARRAY GSSTACK(99),LATSTACK(99)INTSTACK(99) $,
  REAL ARRAY PPGS(16),PGS(16),QGS(4),XKGS(4),
      PPL(16), PL(16), QL(4), XKL(4),
  REAL ARRAY TRLAT(3),TRGS(3),CDUM(4),CDUMG(4) $,
  INTEGER DISCRETE,INPUT,WORD $,
  INTEGER MODE $,
  LOGICAL GSCORT, FIRSTIME $,

```

```

... *****//      BOOLEAN NABLGS $,
      PPSET TRLAT(1) = 0.9753099 $,
      PPSET TRLAT(2) = 0.9048374 $,

... THE FOLLOWING VARIABLES SHARE SPACE IN BFFR //
... UO = BFFR(0),PSIEMHDG=BFFR(1), .... TSINR=BFFR(8),TCOSR=BFFR(9) //

      SYNONYMS BEGIN
            PSIEMHDG = GM $,
            PSIECRS = Z $,
            KSINPHI = KSINOMEGA $,
            KCOSPHI = KCOSOMEGA $,
            KSINPHI = KSINOMEGA $,
            GM = STC1, TQM $,
            HSINR = HPTR.INP $,
            HCOSR = VPTR.INP $,
      END $,

      POINTER COMPONENT WHOLE $,
      WHOLE $$$ 0 $,
      PROCEDURE DWORD.OUT,DIG.OUT $,
      PROCEDURE PTR.FLAG
      REAL PROCEDURE FORINT,KALCLR, KFIL1, KFIL2 $,
      REAL PROCEDURE SQRT, EXP, ATANB, ATAN, SIN, COS $,
      REAL PROCEDURE LIMIT,SIGN $,
... *****//      BOOLEAN CKPT $,
... *****//      PROCEDURE ASMDCC $,
... *****//      PROCEDURE ASMFLO $,
... *****//      PROCEDURE PNT $,

... *****//

... *****//      REAL GH,ATSW,VHTN,ATGSN $,

      SYNONYMS 100C = LBSO.MASK $,

... LOCAL VARIABLES (LOCAL TO THIS STACK ONLY) //
      INTEGER LBSO.MASK,IT $,
      INTEGER FLCT,MODE,INDEX $,
      BOOLEAN VERT,PTR.FLAG,ITEST $,
      REAL RL,LC,PHIC,RA,ODB,CTW,VHG,QL02,RE,AY,QH,KY,GAMH,
      PSINOW,
      PHILAG,
      HPHIC,PHICT,GAME,PHOLD,XSIL,DUMP,LAT.RH,
      GP,GHH,OVSTP,X2L,YDT,YO,VPTR $,
      REAL FK,CTE,WH,QH,SAH,CWH,AED1,AED2 ... STACK VARIABLES $,
... PPSET OR SYNONYMS NEEDED TO DEFINE SELF TEST C CONSTS //
... *****//      REAL C01,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14 $,
      REAL H,QTR.INCH.DISPLACEMENT.VALUE,
      V,QTR.INCH.DISPLACEMENT.VALUE $,
      REAL PH,HPTR $,
... THE ABOVE MUST BE CONSTANTS //
... *****//      PROCEDURE OUTPUT $,
... *****//      PROCEDURE MXPRT $,
... *****//

```

```

      DEFINE PROCEDURE LATPGM TOBE
      BEGIN

```

```

MODE,CONTROL $
      RH = GLBL.RH $,      ...  BUFFER TO SAVE GLOBAL REFERENCE //

      GOTO MODE.SWITCH(MODE) $,
      SWITCH MODE.SWITCH = TACM,STM,MHDGM,ILSM,OFFMD $,

STMS      MODE.INDEX = 7 $,
          GOTO MAIN.COMPUTATION $,
MHDGMS    MODE.INDEX = 4 $,
          IT = 0 $,
          GOTO MHMD $,
ILSMS     IF FIRSTIME THEN BEGIN ILS.INIT() $,
          MODE.INDEX = 4 $, END $,
          ILS.COMMON()      ...  PREPARATORY CALCULATIONS $,
          PTR.FLAG( MODE.INDEX, 15, 13, LOCFLAG() )
          IF LOCFLAG() THEN GOTO MAIN.COMPUTATION $,

          GOTO LOC.SWITCH(MODE.INDEX-3) $,
          SWITCH LOC.SWITCH = MHSUBM,LOCAP,LOCTRK $,

MHSUBM    IF LOC.BEAM.SENSOR.ETC() THEN MODE.INDEX = 5 $,
          GOTO MAIN.COMPUTATION $,
LOCAP     IF LOC.CAP.TO.TRACK() THEN MODE.INDEX = 6 $,
LOCTRK    GOTO MAIN.COMPUTATION $,

TACMS     IF FIRSTIME THEN BEGIN TAC.INIT() $, MODE.INDEX=4 $, END $,
          TAC.COMMON()      ...  PREPARATORY CALCULATIONS $,
          PTR.FLAG( MODE.INDEX, 15, 13, TACFLAG() )
          IF TACFLAG() THEN GOTO MAIN.COMPUTATION $,

          GOTO TACSWITCH(MODE.INDEX) $,
          SWITCH TACSWITCH = TACOVS,TACAP,TACTRK,TACMH $,

TACMHS    IF TACMH.TO.CAP() THEN MODE.INDEX = 2 $,
          GOTO MAIN.COMPUTATION $,
TACAP     IF TACAP.TO.TRK() THEN MODE.INDEX = 3 $,
          GOTO MAIN.COMPUTATION $,
TACOVS    IF OVS.TO.CAP() THEN BEGIN MODE.INDEX = 2 $,
          ... //
          AED1 = PSIECRS $,
          PSIECRS = PSIECRS + PSINOW $,
          ISVSKP( 0.002974243164, 0.9536825019, 1.0,
                  0.0004274698477, 0.8919864872, 0.0003169697243,
                  0.004067843615, 0.0002222222222 ) $,
          YKL(1) = TCM*RH $,
          YKL(2) = -UC*SIN( PSIECRS )/0.9574377611 $,
          X2L = YKL(2) $,
          YKL(3) = YKL(4) = CTW = 0.0 $,
          PSIECRS = GAMH = XSIL = AED1 $,
          END $,
          GOTO MAIN.COMPUTATION $,

```



```
TACTRKS  IF TTRK.TO.OVS() THEN MODE.INDEX = 1 $,
        GOTO MAIN.COMPUTATION $,
```

MAIN.COMPUTATIONS

```
... *****// IF CKPT THEN PRT(0,.C. /MAIN.COMPUTATION /) $,
```

```
        GOTO MCSWITCH(MODE.INDEX) $,
        SWITCH MCSWITCH = TOVSMD,TCMD,TTKMD,MHMD,LCMD,LTKMD,STMD $,
```

```
OFFMDS  DIG.OUT(9,TRUE) ... VPTR OUT OF VIEW $,
        DIG.OUT(8,TRUE) ... HPTR OUT OF VIEW $,
```

```
        GOTO LAT.RET $,
```

```
STMS    SELF.TEST() $,
        OUTPUT(0,VPTR) $,
        OUTPUT(1,HPTR) $,
```

```
        GOTO LAT.RET $,
MHMS    MAN.HDG.COMPUTATION(PSIEMHDG) $,
        GOTO LATERAL.OUT $,
```

```
LCMS    LCAPTURE.COMP() $,
        GOTO LATERAL.OUT $,
```

```
LTKMS    LTRACK.COMP() $,
```

```
LATERAL.OUT$ LAT.FINISH.COMMON() $,
        VPTR = VPTR/0.1591549 $,
        OUTPUT(0,VPTR) $,
```

```
        IF ( IT GET 0 ) THEN KFILL( PL, PPL, TRLAT, QL,
        IF MODE EQL 1 THEN RH ELSE 1.0, RL, SDUM, CDUM ) $,
```

```
        GOTO LAT.RET $,
```

```
TCMS    TACAPTURE.COMP() $,
        GOTO LATERAL.OUT $,
```

```
TOVSMD$ MAN.HDG.COMPUTATION(PSIECRS) $,
        GOTO LATERAL.OUT $, ... NOTE THIS IS HH MODE //
        GOTO TACFINISH $, ... NOTE THIS IS HH MODE //
```

```
TTKMS    TAC.TRACK.COMP() $,
        GOTO LATERAL.OUT $,
```

```
        B
LAT.RET$ DONE = TRUE $,
WAITLP $
```

```
... *****// IF CKPT THEN PRT(0,.C. /WAITLP/) $,
```

```
        STSUB() $,
```

```
... ADD IF DONE THEN GO TO WAITLP $,
```

```
... *****// GO TO MODE.CONTROL $,
```

```
        END .. OF LATERAL PROCEDURE DEFINITION $,
```

```
DEFINE PROCEDURE PTR.FLAG( P.MODE, P.FLAG, FLAG, COND )
```

```
    WHERE INTEGER P.MODE, P.FLAG, FLAG BOOLEAN COND
```

```
    TOBE BEGIN
```

```
        IF ( P.MODE NEQ 1 OR 4 ) THEN FLAG = FLAG OR P.FLAG $,
```

```
        DIG.OUT( FLAG, COND ) $,
```

```
    END $,
```

```
DEFINE PROCEDURE MAN.HDG.COMPUTATION(AED1)
```

```
    WHERE REAL AED1 TOBE
```

```

      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /MAN.HDG,COMP/) $,
... //              GHH = (UO-0.35742857)/0.537142852 $,
... //              GHH = LIMIT(0.3333333333,GHH) + 0.66666666667 $,
... // REAL AED2 $  AED2 = LIMIT( 0.16666667, AED1 ) $,
                  PHIC = GHH*AED2/( -0.3333333333 ) $,
                  DIG.OUT(8,FALSE) ... HPTR OUT OF VIEW BIT $,
                  DIG.OUT(9,TRUE) ... VPTR IN VIEW $,
      END $,

```

DEFINE PROCEDURE ILS.INIT TOBE

```

      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /ILS.INIT/) $,
                  RA = RH = 0.15625 $,
                  AY = 0.1 $,
                  LAT.INIT() $,
                  LC = 0.37136 $,
                  ISVSKP( 0.098304, 0.000000375, 0.0065536,
                        0.0000000238418, 0.0119395325, 0.0001086717952,
                        0.00003917558372, 0.01422222222 ) $,
      END $,

```

... THIS PROCEDURE IS A SAVE-SKIP TYPE //

DEFINE PROCEDURE LAT.INIT TOBE

```

      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /LAT.INIT/) $,
                  KALCLR(XKL,PL) $,
                  RL = 0.0 $,
                  IT = 0 $,
                  QH = QM $,
                  CTW = 0.0 $,
                  VHG = UO $,
                  GAMH = PSIECRS $,
                  PHOLD = PHI $, ... IC BANK RATE COMP. //
                  PHILAG = 0.0 $,
      END $,

```

... THIS PROCEDURE IS A SAVE-SKIP TYPE //

DEFINE PROCEDURE ISVSKP(A,B,C,D,E,F,G,H)

WHERE REAL A,B,C,D,E,F,G,H TOBE

BEGIN

```

... *****//      IF CKPT THEN PRT(0,,C. /ISVSKP/) $,
                  PL(1) = A $,
                  PL(6) = B $,
                  PL(11) = C $,
                  PL(16) = D $,
                  QL02 = E $,
                  QL(3) = F $,
                  QL(4) = G $,
                  TRLAT(3) = H $,
      END $,

```

```

DEFINE PROCEDURE ILS.COMMON TOBE
BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /ILS.COMMON/) $,
      IF ( IT LES 1 ) THEN
      BEGIN
        XKL(1) = QH $,
        XKL(2)=-0.44300*U0*SIN(PSIECRS-0.03125*QH)/RH $,
      END $,
        IF ABS(QH) LEQ 0.6399994
          THEN RA = 0.44300*VHG*SIN(0.03125*QH-GAMH)/QDB $,
          RE = RH = 0.000845*VHG*COS(GAMH) $,
          ... RANGE RATE FILTER . .(RANGE ESTIMATE) //
... // REAL RD $,   RD = RA - RE $,
          IF ( ABS(QH) LT 0.1772950 AND MODE.INDEX
              NEQ 4 ) THEN AY = 0.95*AY $,
          PH = RE + AY*LIMIT(0.0625,RD) $,
      IF GSCORT
        THEN PH = RE + 0.1*( SH = RE + 0.06430288 )
        ELSE IF RH LEQ 0.15625 THEN RH = 0.15625 $,
        GLBL.RH = RH $,
        LAT.COMMON(RH,1.0,QH,-0.03458,0.005945,0.18332,1.0) $,
... LAT.RR7 //
... //      KY = 0.1029506135/(0.110671558+0.8893284419*
                                ABS(QH)) $,
        PHICT = (KY*QH + QDB)/0.6873275443 $,
      END $,

... THIS PROCEDURE IS A SAVE-SKIP TYPE //

DEFINE PROCEDURE LAT.COMMON( RH1, RH2, SQM, KFK, KCTE,
KSCAL1, KSCAL2 ) WHERE REAL RH1, RH2, KFK, KCTE,
KSCAL1, KSCAL2, DQM, SQM, CWH, SWH, $,
TOBE
BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /LAT.COMMON/) $,
        PHILAG = 0.9995*( PHILAG = PHI ) + PHI $,
        PHILAG = LIMIT( 0.00060937, PHILAG ) $,
        PHI = PHI - PHILAG $,
        SINPHI = SIN( PHI ) $,
        COSPHI = COS( PHI ) $,
... LCMN.BR1 //      QL(2) = IF IT LES 25 THEN QL02
                                ELSE 0.00025*QL02 $,
                                ... QL(2) IS DOUBLE PRECISION //
KALMNS      IF (IT GRT 0) THEN BEGIN
        H = RH2 $, ... DOUBLE CONVERSION //
        ZK = SQM $, ... DOUBLE CONVERSION //
... //      FK = KFK*SINPHI*COS(GAMH)/(RH1*COSPHI) $,
        KFIL2(TRLAT,XKL,H,ZK,FK,SDUM,CDUM) $,
        END $,
... *****//      IF CKPT THEN BEGIN
... *****//      MXPRT(.C. / PL MATRIX/, PL,16) $,
... *****//      MXPRT(.C. / XK MATRIX/,XKL, 4) $,
... *****//      END $,
... // REAL QH0 $,   QH0 = QH $, ... BUFFER OLD VALUE //
        QH = XKL(1)/PH2 $,

```



```

... //
EQ7BGS
      ODB = XKL(2) $,
      GAME = GAMH + 0.000845*SINPHI/(VHG*COSPHI) $,
      CTE = RH*( QH0 - QH ) - KCTE*VHG*SIN(GAME) $,
      CTE = LIMIT(0.05142857*KCTE,CTE) $,
      CTW = IF (MODE,INDEX EGL 4)
              THEN 0.0
              ELSE (CTW + CTE/LC) $,
      WH = ABS(CTW) $,
      OH = SIGN(0.5,CTW) $,
      CWH = WH*COS(OH-PSIECRS) $,
      SWH = WH*SIN(OH-PSIECRS) $,
      AED1 = UO + CWH $,
      VHG = SQRT( SWH*SWH + AED1*AED1 ) $,
      GAMH = PSIECRS + ATAN8(SWH,AED1) $,
      AED1 = KSCAL2*VHG*VHG*(0.5-0.5*COS(GAMH)) $,
      AED1 = -SIGN(AED1,GAMH) $,
      AED2 = KSCAL1*PH*ABS(OH) $,
      PHICL = ATAN8(AED1,AED2) $,
END $,

```

```

      DEFINE BOOLEAN PROCEDURE LOC.BEAM.SENSOR.ETC TOBE
      BEGIN
... *****//      IF CKPT THEN PRT(0,.C. /LOC.BEAM.SENSOR.ETC/) $,
      LOC.BEAM.SENSOR.ETC = FALSE $,
      IF DISCRETE.INPUT.WORD .A. LBSO.MASK NEQ 0 THEN GOTO TR $,
... LAT.BR10 //
      RL = 0.0 $,
      IF (ABS(QH) GRT 0.5688885) THEN BEGIN
          IT = 0 $,
          CTW = 0.0 $,
          RH = 0.15625 $,
          GOTO RETURN $,
          END $,
      RL = 0.000016384 $,
      IF (IT LES 30) THEN BEGIN IT = IT + 1 $,
          GOTO RETURN $,
          END $,
      PL = 0.00016384 $,
      IF (RH GRT 0.3125) AND (ABS(QH) GRT 0.426666666) OR
          (RH LEQ 0.3125) AND (ABS(QH) GRT 0.5688885-0.4550904*RH)
          THEN GOTO RETURN $,
      IF ( 0.0711111 GRT ABS(QH) ) THEN GOTO TR $,
      IF ( ABS(PHICL) LES (0.011111111 + 0.0625*ABS(QH)))
          THEN GOTO RETURN $,
      LOC.BEAM.SENSOR.ETC = TRUE $,
      NABLS = TRUE $,
      RL = 0.00016384 $,
      IT = 30 $,
... UNMASK TIMER2 INTERRUPT //
... ADD      DUMP = TRIB(INT.NO) $,
... ADD      W0(MSK.REG) = ( T2MASK .V. W0(MSK.REG)) $,
      END $,

```

```

      DEFINE PROCEDURE LCAPTURE.COMP TOBE
      BEGIN

```

```

... *****//      IF CKPT THEN PRT(0,,C. /LCAPTURE,COMP/) $,
                     PHIC = PHICL $,
                     END $,

```

```

DEFINE BOOLEAN PROCEDURE LOC.CAP.TO.TRACK TOBE
BEGIN

```

```

... *****//      IF CKPT THEN PRT(0,,C. /LOC.CAP.TO.PRETRK/) $,
                     IF ( ( 0.07111111 GRT ABS(QH) ) OR
                     ( 0.1066465 GRT ABS(QH) AND 0.11111112 GRT ABS(GAMH) )
                     OR ( -0.1066465 GRT QH AND GAMH GRT 0.11111112 )
                     OR ( QH GRT 0.1066465 AND -0.11111112 GRT GAMH ) )
                     THEN BEGIN
                     LOC.CAP.TO.TRACK = TRUE $,
                     HPHIC = PHIC - PHICT $,
                     RL = 0.0016384 $,
                     END
                     ELSE LOC.CAP.TO.TRACK = FALSE $,
                     END $,

```

```

DEFINE PROCEDURE LTRACK.COMP TOBE
BEGIN

```

```

... *****//      IF CKPT THEN PRT(0,,C. /LPRETRK,COMP/) $,
                     AY = 0.8824969026*AY $,
                     PHIC = HPHIC + PHICT $,
                     HPHIC = 0.985*HPHIC $,
                     IF RH LEQ 0.15625 THEN PHIC = RH*PHIC/0.15625 $,
                     END $,

```

```

DEFINE PROCEDURE LAT.FINISH.COMMON TOBE
BEGIN

```

```

... *****//      IF CKPT THEN PRT(0,,C. /LAT.FINISH.COMMON/) $,
... // REAL PHII $, PHII = PHIC - PHOLD $,
                     PHIC = PHOLD + LIMIT(0.00277757,PHII) $,
                     PHIC = LIMIT(0.1687042,PHIC) $,
                     PHOLD = PHIC $,
                     VPTR = PHIC - PHI $,
                     END $,

```

```

DEFINE PROCEDURE TAC.INIT TOBE
BEGIN

```

```

... *****//      IF CKPT THEN PRT(0,,C. /TAC.INIT/) $,
                     LAT.INIT() $,
                     RL = 0.0305175 $,
                     PSINOW = GAMH $,
                     QH = TQM $,
                     LC = 0.2321 $,
                     ISVSKP( 0.002974243164, 0.9536825019, 1.0,
                     0.0004274698477, 0.8919864872, 0.0003169697243,
                     0.004067843615, 0.000222222222 ) $,

```

END \$,

DEFINE PROCEDURE TAC.COMMON TOBE
BEGIN

```
... *****//      IF CKPT THEN PRT(0, .C. /TAC.COMMON/) $,  
... TAC.BR1 // IF ( IT LES 1 ) THEN  
      BEGIN  
        XKL(1) = TQM*RH $,  
        XKL(2) = -U0*SIN(PSIECRS)/0.9574377611 $,  
        X2L = XKL(2) $,  
        XSIL = GAMH $,  
      END $,  
... TAC.BR2 //      LAT.COMMON(1.0, RH, TQM, -0.08332875458, 0.0002321,  
        1.0, 0.190603125 ) $,  
... GP = 0.09765625/RH //  
      IF (RH LES 0.09765625) THEN GP = 1.0  
        ELSE GP = 0.09765625/RH $,  
      OVSTP = 0.0328125*U0 $,  
      X2L = 0.9801454*( X2L = XKL(2) ) + XKL(2) $,  
      ... 1ST WORD OF XKL(2) //  
      XSIL = 0.9801454*( XSIL = GAMH ) + GAMH $,  
      YDT = X2L = VHGX(GAMH = XSIL)/0.3047619047 $,  
      PHICT = GP*(XKL(1)+0.192*YDT)/0.3580985 $,  
    END $,
```

DEFINE BOOLEAN PROCEDURE TACMH.TO.CAP TOBE
BEGIN

```
      REAL ADJPHL $,  
... *****//      IF CKPT THEN PRT(0, .C. /TACMH.TO.CAP/) $,  
      IF DISCRETE.INPUT.WORD .A. LBSO.MASK NEQ 0  
        THEN GOTO TRU $,  
      TACMH.TO.CAP = FALSE $,  
... TAC.BR5 // IF ( (RH*ABS(TQM) GRT 0.099471839)  
      AND (IT LES 1) ) THEN GOTO RETURN $,  
      IF (IT LES 30 )  
        THEN BEGIN  
          IT = IT + 1 $,  
          GOTO RETURN $,  
        END $,  
... TAC.BR6A //      ADJPHL = 0.35 + LIMIT(0.15, (RH=0.136725)/0.52083) $,  
... //      IF ( (PHICL*PHICT LES 0.0) OR (ABS(PHICT) LES  
        ABS(PHICL*ADJPHL)) ) OR  
        ( (ABS(PHICT) GRT 0.00555769) AND  
          (ABS(RH) LES 0.0888889 ) ) THEN  
          GOTO RETURN $,  
      TACMH.TO.CAP = TRUE $,  
      CTX = 0.0 $,  
      IT = 30 $,  
    END $,
```

DEFINE PROCEDURE TACAPTURE.COMP TOBE
BEGIN

```
... *****//      IF CKPT THEN PRT(0, .C. /TACAPTURE.COMP/) $,
```



```

RL = 0.0305175 $,
PSINOW = PSIECRS $,
IF (ABS(PHICL) LES ABS(PHICT)) AND
  ( 0.0 GR GH*PHICT )
  THEN BEGIN
    PHIC = PHICL $,
    ITEST = TRUE $,
    END
  ELSE BEGIN
    PHIC = PHICT $,
    ITEST = FALSE $,
    END $,

```

```

END $,

```

```

      DEFINE BOOLEAN PROCEDURE TACAP.TO,TRK TOBE
... *****//      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /TACAP.TO,TRK/) $,
      TACAP.TO,TRK =
      (NOT ITEST) AND (ABS(GH) LEQ 0.0888889 ) $,
... *****//      END $,

```

```

      DEFINE PROCEDURE TAC,TRACK,COMP TOBE
... *****//      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /TAC,TRACK,COMP/) $,
      PHIC = PHICT $,
      RL = LIMIT(0.1525878906,0.78125*RH) $,
      PSINOW = 0.90483741*( PSINOW-PSIECRS ) + PSIECRS $,
... *****//      END $,

```

```

      DEFINE BOOLEAN PROCEDURE OVS.TO,CAP TOBE
... *****//      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /OVS.TO,CAP/) $,
      OVS.TO,CAP = RH GEQ OVSTP $,
... *****//      END $,

```

```

      DEFINE BOOLEAN PROCEDURE TTRK.TO,OVS TOBE
... *****//      BEGIN
... *****//      IF CKPT THEN PRT(0,,C. /TTRK.TO,OVS/) $,
      TTRK.TO,OVS = RH LES OVSTP $,
... *****//      END $,

```

```

      DEFINE BOOLEAN PROCEDURE TACFLAG TOBE
      TACFLAG = (DISCRETE.INPUT,WORD .A. 200C) NEQ 0 $,

```

```

      DEFINE BOOLEAN PROCEDURE LOCFLAG TOBE
      LOCFLAG = (DISCRETE.INPUT,WORD .A. 400C) NEQ 0 $,

```

DEFINE PROCEDURE FAILURE.WARNING TOBE

BEGIN DIG.OUT(10,TRUE) \$,
DIG.OUT(14,TRUE) \$,
DIG.OUT(15,TRUE) \$,
END \$,

SYNONYMS BEGIN

314630 = C01 \$,
125250 = C2 \$,
.875 = C3 \$,
.625 = C4 \$,
400010 = C5 \$,
.0628 = C11 \$,
.00628 = C12 \$,
.0628 = C13 \$,
.00628 = C14 \$,
.5 = C7 \$,
.1 = C8 \$,

END \$,

DEFINE PROCEDURE STSUB TOBE BEGIN

... IF ((C01+C2)/C3)*C4-C5 NEQ C6 THEN FAILURE.WARNING() \$,
GOTO RETURN \$,
FAILURE.WARNING() \$,
END \$,

DEFINE PROCEDURE SELF.TEST TOBE

BEGIN
IF NOT FIRSTIME THEN BEGIN
IF ABS(VPTR.INP-C11) GRT C12 THEN FLCT = FLCT+1 \$,
IF ABS(HPTR.INP-C13) GRT C14 THEN FLCT = FLCT+1 \$,
END ELSE FLCT = 0 \$,
IF ABS(STC1 -C7) GRT C8 THEN FLCT =FLCT+1 \$,
HPTR = H.QTR.INCH.DISPLACEMENT.VALUE \$,
VPTR = V.QTR.INCH.DISPLACEMENT.VALUE \$,
IF FLCT GRT 5 THEN FAILURE.WARNING() \$,
END \$,

END FINI

SENTINEL CARD FOR U-1108 TERMINAL

JFM AFD, DFDPGM,DFDPGM
BEGIN

... THE FOLLOWING IS THE GLIDESLOPE PROGRAM BLOCK //

... *****//

... GLOBAL VARIABLES (COMMON TO ALL 3 STACKS) //
REAL PSIEHPOG,PSIECRS,KSINPHI,KCOSPHI,QM,HSINR,HCOSR,TSINR,

```

USINR,UCOSR,
    TCOSR,SINPHI,COSPHI,RH,UO,TM,SDUM,SDUMG,PHI    $,

REAL QL02 $,

COMMON GSSTACK,LATSTACK,INTSTACK,DISCRETE,INPUT,WORD,FIRSTIME,
    UO,GM,Z,KSINOMEGA,KCOSOMEGA,GM,HSINR,HCOSR,
    TSINR,TCOSR,USINR,UCOSR,PPGS,PGS,QGS,XKGS,
    PPL,PL,QL,XKL,XKPL,CDUM,CDUMG,GSCORT,
    SINPHI,COSPHI,RH,UO,TM,SDUM,SDUMG,MODE,
    $,
    REAL ARRAY GSSTACK(99),LATSTACK(99),INTSTACK(99) $,
... *****// ,CKPT,GM,WH,VHG,GAMH,GH,    ATSW,VHTN,ATGSN ,PHI,NARLGS

REAL ARRAY  PPGS(16),PGS(16),QGS(4),XKGS(4),
    PPL(16),PL(16),QL(4),XKL(4) $,
REAL ARRAY TRLAT(3),TRGS(3),CDUM(4),CDUMG(4) $,
INTEGER DISCRETE,INPUT,WORD $,
INTEGER MODE $,
BOOLEAN GSCORT, FIRSTIME $,
... *****//    BOOLEAN NARLGS $,
    PRESET TRLAT(1) = 0.9753099    $,
    PRESET TRLAT(2) = 0.9048374    $,
PRESET TRGS(3) = 0.0142222222222 $,

... THE FOLLOWING VARIABLES SHARE SPACE IN BFFR //
... UO = BFFR(0),PSIEMHOG=BFFR(1), .... TSINR=BFFR(8),TCOSR=BFFR(9) //

SYNONYMS BEGIN
    PSIEMHOG = GM    $,
    PSIECRS = Z    $,
    KSINPHI = KSINOMEGA $,
    KCOSPHI = KCOSOMEGA $,
    KSINPHI = KSINOMEGA $,
    GM = STC1, TOM $,
    HSINR = HPTR.INP $,
    HCOSR = VPTR.INP $,
END $,

POINTER COMPONENT WHOLE $,
WHOLE $=$ 0 $,
PROCEDURE PTR.FLAG
REAL PROCEDURE FORINT,KALCLR, KFIL1, KFIL2 $,
PROCEDURE DWORD.OUT,DIG.OUT $,
REAL PROCEDURE SORT, EXP, ATANB, ATAN, SIN, COS $,
REAL PROCEDURE LIMIT,SIGN $,
... *****//    BOOLEAN CKPT $,
... *****//    PROCEDURE ASMDEC $,
... *****//    PROCEDURE ASMFLO $,
... *****//    PROCEDURE PRT $,

... *****//

... LOCAL VARIABLES (LOCAL TO THIS STACK ONLY) //

... *****//    REAL GM,WH,VHG,GAMH $,
    REAL UF1,UF2,TC,HPTR,TDC,ATSW,VHTN,ATGSN,GDH,TF1,TF15,TDH,SHO,

```



```

        GND,TF1SC,TDCAT,GDF $,
REAL GH $,
REAL FKGS,TDCA,TDCAV ... STACK VARIABLES ONLY $,
INTEGER VRSD,MASK $,
INTEGER GS,INDEX, GS.IT $,
... RG IS DOUBLE PRECISION //
REAL ARRAY RG(3) $,
PRESET RG(1) = 0.000000078125 $,
PRESET RG(2) = 0.00000015625 $,
PRESET RG(3) = 0.000003125 $,
PRESET TRGS(1) = 0.09752456 ... EXP(-.025) $,
PRESET TRGS(2) = 0.09511125 ... EXP(-.05) $,
PRESET TRGS(3) = 0.01422222222 $,

... *****//

        DEFINE PROCEDURE GSPGM TOBE
BEGIN
ILS4PM $ PTR.FLAG( GS,INDEX 10, 0, R.ALT.FLAG() ) $,
        ... RADAR ALTITUDE VALID $,
PTR.FLAG( GS,INDEX, 14, 12, GSFLAG() )
IF GSFLAG() THEN GOTO MAIN.COMPUTATION $,

GOTO GSSWITCH(GS,INDEX) $,
SWITCH GSSWITCH = GSPC,GSCAP,GSTRK,GSIDL $,

GSIDL$ GSTEST() $, GOTO MAIN.COMPUTATION $,
GSCAP $ IF GS.C.TO.TRK() THEN GS.INDEX = 3 $,
        GOTO GSTRK $,
GSPC $ IF GS.P.TO.CAP() THEN BEGIN GS.INDEX =2 $, GSCORT=TRUE $,
        TC = TM $,
        DIG.OUT(11,FALSE) ... GPENGAGE $,
        DIG.OUT(8,FALSE) ... HPTR IN VIEW $, END $,
GSTRK$ GS.COMMON() $, GOTO MAIN.COMPUTATION $,

MAIN.COMPUTATIONS

        SWITCH GS.COMP.SWITCH = GSPCMD,GSCMD,GSTKMD,GS.RET $,
        GOTO GS.COMP.SWITCH(GS.INDEX) $,

GSCMD$ GS.CAP.COMP() $,
        GOTO GSFINISH $,
GSPCMD$ GS.PRECAP.COMP() $,
        GOTO GSFINISH $,
GSTKMD$ GS.TRACK.COMP() $,
GSFINISH$
        HPTR = HPTR/0.1591549 $,
        OUTPUT( 1, HPTR ) $,
        KFILI(PGS,PPGS,TRGS,GGs,1,0,RG(GS.INDEX),SDUMG,CDUMG) $,
GS.RET $ DONE = TRUE $,
... ADD HALT $,

END ... OF GS PROC $,

```

```

DEFINE PROCEDURE GSTEST TOBE
  IF GM LES 0.3982222222222 AND NARLGS
    THEN BEGIN GS.INDEX = 1 $,
      GS.INIT() $,
    END $,

```

```

DEFINE PROCEDURE GS.INIT TOBE
  BEGIN

```

```

... *****//
  IF CKPT THEN PRT(0,,C. /GS.INIT/) $,
  GS.IT = 0 $,
  GH = GM $,
  SH = 0.45836625*Z/( 0.64 - 0.36*GH ) $,
  GDH = -0.3655590099*U0*SIN(ATGSN=0.01388888847+
    TM ))/SH $,
  VHTN = UF1 = U0 $,
  ATSW = UF2 = 0.0 $,
  GDF = GDH $,
  TF1 = TF15 = TM $,
  KALCLR(XKGS,PGS) $,
  PGS(1) = 0.1090688 $,
  PGS(6) = 0.01378051757 $,
  PGS(11) = 0.1090688 $,
  PGS(16) = 0.0001355074364 $,
  QGS(2) = 0.0002328737791 $,
  QGS(3) = 0.0006379424 $,
  QGS(4) = 0.002524766387 $,
  XKGS(1) = 0.25*GM $,
  XKGS(2) = GDH $,
  END $,

```

```

DEFINE PROCEDURE GS.COMMON TOBE
  BEGIN

```

```

... *****//
... GS.BR3 //
  IF CKPT THEN PRT(0,,C. /GS.COMMON/) $,
  IF GSCORT THEN A15=0.99335555 ELSE A15=0.9048374 $,
  TF15 = A15*(TF15-TM)+TM $,
  TF15C = TM - TF15 $,
  UF1 = 0.98420*( UF1 - U0 ) + U0 $,
... // REAL UF1C $, UF1C = U0 - UF1 $,
  UF2 = 0.984210488*( UF2 - UF1C ) + UF1C $,
... // REAL SH0 $, SH0 = SH $,
  SH = 0.9*SH = 0.00065625*VHTN*COS(ATGSN) +
    0.045836625*Z/( 0.64 - 0.36*GH ) $,
  IF ( 0.0 GRT SH ) THEN SH = 0.0 $,
  TF1 = 0.9048374*( TF1 - TM ) + TM $,
  TDM = TM-TF1 $,
... //
... // REAL ATGSE $, ATGSE = ATGSN + 0.1*TDM $,
... //
  AED1 = SH = SH0 + 0.000845*VHTN*COS(ATGSE) $,
  ATSW = ATSW = AED1 $,
  VHTN = U0 + ATSW $,
  IF ( 1 GRT GS.IT ) THEN GOTO GS.BR4 $,
... //
  FK = -VHTN*TDM/(0.1733756*SH) $,
  ... FK IS LOCAL, DOUBLE //

```

```

REAL H,ZK $,
      H = 1.0 $, ... H IS LOCAL, DOUBLE //
      ZK = 0.25*GM $, ... ZK IS LOCAL, DOUBLE //
      KFIL2(TRGS,XKGS,H,ZK,FK,SDUMG,CDUMG) $,
GS,HR4 $
... *****//
... *****//
... *****//
... *****//
      GS.IT = 1 $,
      IF CKPT THEN BEGIN
        MXPRT(.C. / PGS MATRIX/,PGS,16) $,
        MXPRT(.C. / XKGS MATRIX/,XKGS, 4) $,
      END $,
      GH = XKGS(1)/0.25 $,
      GDH = XKGS(2) $,
      GDF = 0.9*( GDF - GDH ) + GDH $,
      ATGSN = -0.6095238095*GDF*SH/VHTN $,
      END $,

      DEFINE PROCEDURE GS.PRECAP.COMP TOBE
      BEGIN
... *****//
      IF CKPT THEN PRT(0,.C. /GS.PRECAP.COMP/) $,
      DIG.OUT(8,TRUE) ... HPTR OUT OF VIEW BIT $,
      END $,

      DEFINE BOOLEAN PROCEDURE GS.P.TO.CAP TOBE
      BEGIN
... *****//
      IF CKPT THEN PRT(0,.C. /GA.P.TO.CAP/) $,
      GS.P.TO.CAP = GH LEQ 0.19911033
      OR DISCRETE.INPUT.WORD .A. VBSO.MASK NEQ 0 $,
      END $,

      DEFINE PROCEDURE GS.CAP.COMP TOBE
      BEGIN
... *****//
... //
      IF CKPT THEN PRT(0,.C. /GS.CAP.COMP/) $,
      TDC = -0.02314*VHTN*( 0.5 - 0.5*COS(ATGSN) )/
      SIGN( SH*GH, ATGSN ) $,
      TC = TC + TDC $, ... INTEGRATE THETA DOT //
      HPTR = (TC - TH)/0.25 = UF2/0.18699956 $,
      END $,

      DEFINE BOOLEAN PROCEDURE GS.C.TO.TRK TOBE
      BEGIN
... *****//
... //
      IF CKPT THEN PRT(0,.C. /GS.C.TO.TRK/) $,
      IF ( GH LEQ 0.0813773 ) THEN
        BEGIN GS.C.TO.TRK = TRUE $,
          TDCA = TDCA.SUB() $,
          TDCAT = HPTR - TDCA $,
        END
      ELSE GS.C.TO.TRK = FALSE $,
      END $,

... THIS PROCEDURE IS A SAVE-SKIP TYPE //

      DEFINE REAL PROCEDURE TDCA.SUB TOBE
      BEGIN

```



```

... //      REAL TDCAV $,
... //      TDCAV = (1.0-VHTN/0.4285714)*GH/0.3775597 $,
... //      TDCAV = TDCAV +
      (0.8832794*GDH+0.4416398*GDF)/VHTN $,
      TDCA.SUB = TDCAV*SH = TF15C $,
END $,

```

```

      DEFINE PROCEDURE GS.TRACK.COMP TOBE
      BEGIN
... *****//      IF CKPT THEN PRT(0,.C. /GS.TRACK.COMP/) $,
... //      TDCA = TDCA.SUB() $,
      HPTR = TDCA + TDCAT $,
      TDCAT = 0.985*TDCAT $,
END $,

```

```

      DEFINE BOOLEAN PROCEDURE GSFLAG TOBE
      GSFLAG = (DISCRETE.INPUT.WORD .A. 1000C) NEQ 0 $,

```

```

      DEFINE BOOLEAN PROCEDURE R.ALT.FLAG() TOBE
      R.ALT.FLAG = DISCRETE.INPUT.WORD .A. 2000C EQL 0 ) $,

```

END FINI

SENTINEL CARD FOR U-1108 TERMINAL

```

@JFN AED,* KALFIL,KALFIL
BEGIN

```

```

      DEFINE PROCEDURE KFIL1(P,PP,TRANS,Q,H,R,SDUM,CDUM)
      WHERE REAL ARRAY P,PP,TRANS,Q,CDUM $,
      REAL H,R,SDUM
      TOBE BEGIN
      REAL ARRAY DUM(16) $,
      INTEGER I,J $,
      KMATRIX(DUM,TRANS,P) $,
      KMATRIX(PP,TRANS,DUM) $,
      PP(6) = PP(6) + Q(2) $,      ... PP(2,2) //
      PP(11) = PP(11) + Q(3) $,      ... PP(3,3) //
      PP(16) = PP(16) + Q(4) $,      ... PP(4,4) //
      FOR I = 1 STEP 1 UNTIL 4 DO
      CDUM(I) = PP(4*(I-1)+1)+H*PP(4*(I-1)+3) $,
      SDUM = CDUM(1) + CDUM(3)*H + R $,
      FOR I = 1 STEP 1 UNTIL 4 DO
      FOR J = 1 STEP 1 UNTIL 4 DO
      P(4*(I-1)+J) = PP(4*(I-1)+J) -
      (CDUM(I)*(PP(J)+PP(8+J)*H))/SDUM $,
      END $,

```

```

      DEFINE PROCEDURE KFIL2(TRANS,XK,H,ZK,FK,SDUM,CDUM)
      WHERE REAL ARRAY TRANS,XK,CDUM $,
      REAL H,ZK,FK,SDUM
      TOBE BEGIN
      REAL ARRAY XKP(4) $,

```

```

INTEGER I $,
REAL SDUM3 $, ... LOCAL TO KFIL2 //
XKP(1) = XK(1) + TRANS(3)*XK(2) $,
XKP(2) = XK(2) + 0.028444444444*(XK(4)+FK) $,
XKP(3) = TRANS(1)*XK(3) $, ... TRANS(3,3) //
XKP(4) = TRANS(2)*XK(4) $, ... TRANS(4,4) //

SDUM3 = H*(ZK-XKP(3)) - XKP(1) $,
FOR I = 1 STEP 1 UNTIL 4 DO
    XK(I) = XKP(I) + CDUM(I)*SDUM3/SDUM $,
END $,

```

... THIS PROCEDURE GENERATES TRANSPOSE(TRANS) \$,

```

DEFINE PROCEDURE KMATRIX(RES,TR,M)
WHERE REAL ARRAY RES,TR,M TOBE
BEGIN
    RES( 1) = M( 1) + TR(3)*M( 5) $,
    RES( 5) = M( 2) + TR(3)*M( 6) $,
    RES( 9) = M( 3) + TR(3)*M( 7) $,
    RES(13) = M( 4) + TR(3)*M( 8) $,
    RES( 2) = M( 5) + 0.028444444444*M(13) $,
    RES( 6) = M( 6) + 0.028444444444*M(14) $,
    RES(10) = M( 7) + 0.028444444444*M(15) $,
    RES(14) = M( 8) + 0.028444444444*M(16) $,
    RES( 3) = TR( 1) * M( 9) $,
    RES( 7) = TR( 1) * M(10) $,
    RES(11) = TR( 1) * M(11) $,
    RES(15) = TR( 1) * M(12) $,
    RES( 4) = TR( 2) * M(13) $,
    RES( 8) = TR( 2) * M(14) $,
    RES(12) = TR( 2) * M(15) $,
    RES(16) = TR( 2) * M(16) $,
END $,

```

```

DEFINE PROCEDURE KALCLR(XK,P)
WHERE REAL ARRAY XK,P TOBE
BEGIN
    INTEGER I,J $,
    FOR I = 1 STEP 1 UNTIL 4 DO
        BEGIN
            XK(I) = 0.0 $,
            FOR J=1 STEP 1 UNTIL 4 DO
                P(4*(I-1)+J)) = 0.0 $,
            END $,
        END $,
    END $,

```

END FINI

SENTINEL CARD FOR U-1108 TERMINAL

```

*JFN AED MAXCNT,MAXCNT
BEGIN

```

```

... SIMULATION CONTROL //
... NOTE MAXCNT MUST BE LEQ 100*KMAX ( ) //

```



```

      IF (ICONTR.LT.1) GO TO 5
      PIE = 3.141592653
      XX = 2.0
      G=58625.3679
      GSA=0.04363323
      T=0.05/(4.0*3600.0)
      A(1)=0.0
      A(2)=0.0
C ** IC GEOMETRY
      DO 1 I=1,9
        XDO(I)=0.0
      1  CONTINUE
      DO 2 I=1,15
        X(I)=0.0
      2  CONTINUE
C ** ILS GEOMETRY
      X(3)=-45.0/57.296
      X(4)=8.5
      X(5)=-0.6
      X(9)=0.21817
      X(10)=125.0
      X(11)=125.0
      3  IF (MODE.NE.4) GO TO 4
C ** TACAN GEOMETRY
      XX = 1.0
      X(3)=45.0/57.296
      X(4)=70.0
      X(5)=7.5
      X(9)=2.0
      X(10)=500.0
      X(11)=500.0
      4  IF (MODE.NE.1) GO TO 51
C ** HEADING HOLD MODE
      Y(3)= 45.0/57.296
      X(4)=150.0
      X(5)=50.0
      X(9)=2.0
      X(10)=125.0
      X(11)=125.0
      51 X3=X(3)
      5  CONTINUE
C ** COMPUTE AIRCRAFT RESPONSE
      DO 6 J=1,4
        XD(1) = (-1.4*X(1)+3600.0*A(1)*PIE)*3600.0
        XD(2)=X(1)
        XD(3) = G*SIN( X(2) )/( X(11)*COS( X(2) ) )
        XD(4)=-X(11)*COS(X(3))
        XD(5)=-X(11)*SIN(X(3))
        XD(6) = (-2.0*X(6)+3600.0*A(2)*PIE)*3600.0
        XD(7)=X(6)
        XD(8)=+3600.0*X(11)*X(6)/SQRT(X(5)**2+X(9)**2)
        XD(9)=X(10)*SIN(X(7))
C ** CALCULATE INTERGAL
      DO 6 I=1,9
        X(I)=X(I)+T*(XD(I)+XDO(I))/2.0
        XDO(I)=XD(I)
      6  CONTINUE

```

```

C ** UP DATA REST OF X VECTOR
  X(12)=X(5)/X(4)
  X(13)=GSA-X(9)/(X(4)-1.5)

  A(9)=SIN(X(7))
  A(10)=COS(X(7))
  A(11)=SIN(X(2))
  A(12)=COS(X(2))
C ** DO NOT NEED TO CHANGE A(13-16) UNTIL UP-DATE DME RANGE
  A(13)=SQRT(X(4)**2+X(5)**2)
  A(14)=0.0
  A(15)=0.0
  A(16)=0.0
  A(17) = 2.0*X(3)/PIE
  A(18) = 2.0*(X(3)-X3)/PIE
  A(19) = XY*16.0*X(12)/PIE
  A(20) = 16.0*X(12)/PIE
  A(21) = 128.0*X(13)/PIE
  A(22) = X(10)/840.0
  A(23) = X(10)/840.0
  A(24) = X(9)/1.0

C   A(25) = A(2)
C   A(25) = X(7)/PIE
C   A(26) = A(1)
C   A(26) = X(2)/PIE

ICONTR=0
END

```

SENTINEL CARD FOR U-1108 TERMINAL

APPENDIX F

TEST EQUIPMENT

APPENDIX F

TEST EQUIPMENT

In order to verify and maintain the 562A-14, two test equipment units were developed at Collins during the DFDC program. These two units are utilized to simulate the airborne electronic environment of the 562A-14 and as a means to troubleshoot the system hardware and software.

The 971D-1 Digital Flight Director Test Set (CPN 622-2022-001) shown in Figure J-1 provides all the inputs which the 562A-14 receives when integrated into the flight control system and serves as an output signal monitor. This test set acts as a stand alone test unit when the 562A-14 is operating on internal memory and is utilized with the 971S-1 CAPS Test Bench for hardware/software tests. The acceptance test procedure for the 562A-14 is performed utilizing the 971D-1 alone. Characteristics of the 971D-1 are summarized below.

I. Outputs:

- Roll Attitude
- Pitch Attitude
- Heading Error
- Course Error
- Glideslope Deviation
- Glideslope Flag
- Localizer Deviation
- Localizer Flag
- Tacan Deviation
- Tacan Flag
- Radar Altitude
- Radar Altitude Flag
- DME Distance
- Airspeed
- Mode Discretes

II. Monitors and Displays:

- Bank Steering Pointer
- Bank Steering Pointer Flag
- Pitch Steering Pointer
- Pitch Steering Pointer Flag
- Glideslope Deviation
- Glideslope Flag
- Glidepath Engage
- Lateral Deviation
- Lateral Deviation Flag
- Flight Director Valid

The 971S-1 CAPS Test Bench shown in Figure J-2 is a minicomputer controlled facility which provides a development and maintenance environment for the CAPS processors. This test system was designed and utilized during the Digital Flight Director program and will be available for maintenance of the 562A-14. The bench can supply the 562A-14 processor's control store, stack memory, and instruction store. Via the controlling minicomputer and peripherals, these memories can be loaded, interrogated, and altered. Additionally, a number of displays and controls are provided to aid the testing process. The test bench is composed of the six major components listed below.

1) PDP-11/05 Processor

On-line control of the test bench is provided by the PDP-11/05 processor. The resident executive software includes the programming necessary to control all peripherals attached to the PDP-11 Unibus.

2) ASR-33 Teletype

The user's interface to the executive program is via directives entered on the teletype keyboard. In addition, the teletype provides paper-tape I/O capability.

3) HP-7970B Magnetic Tape Unit

The HP-7970B provides 9-channel magnetic tape I/O.

4) PICO 1011 Tape Controller

The tape controller provides the necessary formatting, parity, and control electronics to interface the PDP-11/05 processor to the magnetic tape unit.

5) Micromemory 3000 Core Memory

The Micromemory 3000 provides 8K (expandable to 32K) words of 16-bit core memory. It can be used as the stack memory and/or the instruction store for the CAPS processor under test. Additional memory units can be attached to the CAPS Transfer Bus for expanded storage capability.

6) CAPS Test Adapter

The CAPS Test Adapter is a Collins-built peripheral which is attached to the PDP-11 Unibus. Its primary functions are to interface the CAPS Transfer Bus and CAPS Control-Store Bus to the PDP-11 Unibus.

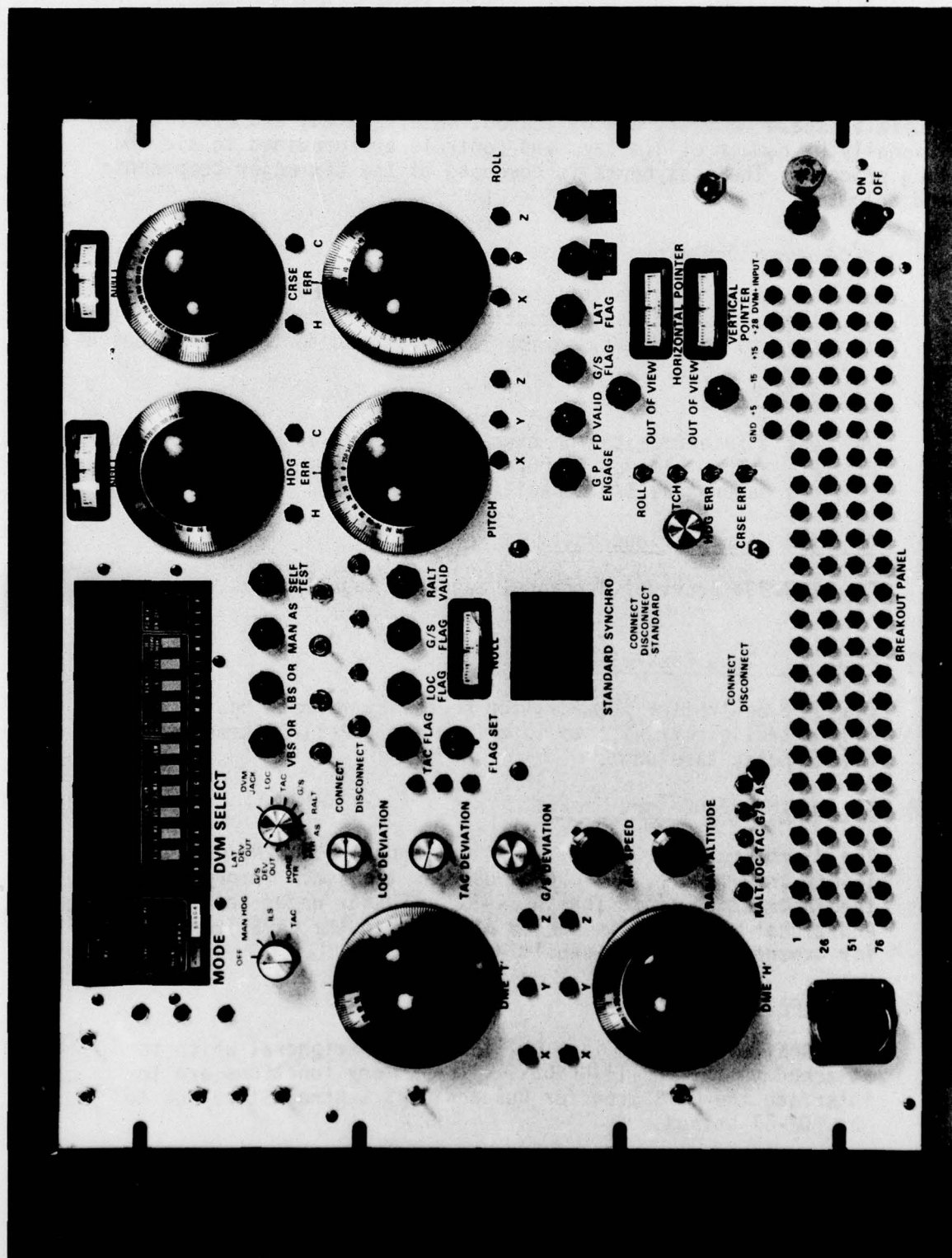


Figure 131 9710-1 Digital Flight Director Test Set

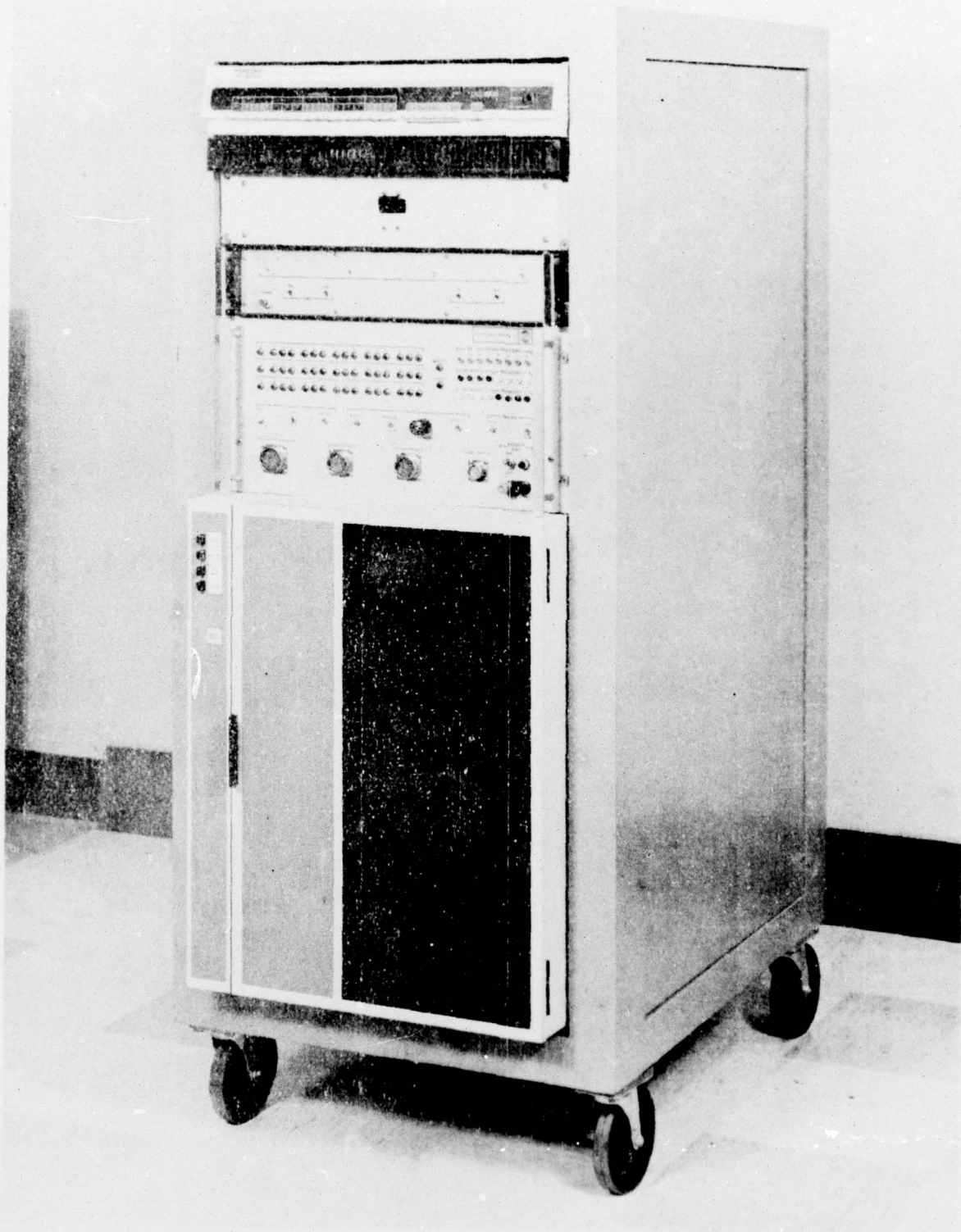


Figure 132 971S-1 CAPS Test Bench
315